IMPLEMENTASI ALGORITMA *WALL FOLLOWING* PADA MANUVER ROBOT KRPAI *QUADRUPED OMNI DIRECTION* MENGGUNAKAN METODE FUZZY SUGENO

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
M. Yusuf Hidayat
NIM: 145150300111043

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI ALGORITMA *WALL FOLLOWING* PADA MANUVER ROBOT KRPAI *QUADRUPED OMNI DIRECTION* MENGGUNAKAN METODE FUZZY SUGENO

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
M. Yusuf Hidayat
NIM: 145150300111043

Skripsi ini telah diuji dan dinyatakan lulus pada
07 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T, M.T	Rizal Maulana , S.T., M.T., M.Sc.
NIK: 19820125 201504 1 002	NIK: 2016078910091001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

IDENTITAS TIM PENGUJI

Penguji 1:

Nama : Dahnian Syauqy,S.T., M.T., M.Sc.

NIP/NIK : 2016078704231000

Penguji 2:

Nama : Gembong Edhi Setyawan, S.T., M.T.

NIP/NIK : 201208 761201 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang,

M. Yusuf Hidayat

NIM: 145150300111043

DAFTAR RIWAYAT HIDUP

Nama	: Muhammad Yusuf Hidayat
Tempat / Tanggal Lahir	: Tulungagung, 28 Maret 1995
Agama	: Islam
E-mail	: tdl.yus@gmail.com
No. Telp/HP	: 081511117656

Riwayat Pendidikan

SD

Tahun Lulus	: 2008
Nama Institusi	: SDN 1 Sumberdadap
Kota Institusi	: Tulungagung

SLTP

Tahun Lulus	: 2011
Nama Institusi	: MTsN Pucanglaban
Kota Institusi	: Tulungagung

SLTA

Tahun Lulus	: 2014
Nama Institusi	: MAN Tulungagung 1
Kota Institusi	: Tulungagung

Perguruan Tinggi

Tanggal Lulus	: 7 Juni 2018
Jurusan/Program Studi	: S1 Teknik Komputer
Nama Perguruan Tinggi	: Universitas Brawijaya
Kota Perguruan Tinggi	: Malang

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan atas rahmat dan karunia-Nya Tuhan yang Maha Esa, sehingga penulis dapat menyelesaikan skripsinya dengan judul "*IMPLEMENTASI ALGORITMA WALL FOLLOWING PADA MANUVER ROBOT KRPAI QUADRUPED OMNI DIRECTION MENGGUNAKAN METODE FUZZY SUGENO* ".

Penulis menyadari bahwa pembuatan skripsi ini tidak lepas dari bantuan baik motivasi, doa dan jasa dari berbagai pihak. Maka sebab itu, penulis menyampaikan rasa hormat dan terimakasih kepada:

1. Kedua orang tua dan seluruh keluarga yang selalu memberikan motivasi dan semangat serta doa sehingga penulis dapat menyelesaikan skripsinya.
2. Bapak Wijaya Kurniawan, S.T, M.T selaku dosen pembimbing pertama yang telah memberikan dukungan dan bimbingannya kepada penulis untuk segera menyelesaikan skripsi ini.
3. Bapak Rizal Maulana , S.T., M.T., M.Sc selaku dosen pembing kedua yang telah memberikan dukungan dalam menyelesaikan laporan skripsi ini.
4. Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
5. Teman-teman robotiik Hendriawan, Agas, Haqqi, Imam Ghozali, Teza Rangga, Hafizhudin, serta semua teman-teman robotiik angkatan 2012 hingga angkatan 2017 yang tidak dapat disebutkan satu persatu.
6. Teman-teman grup dota Jauhar, Hima, Aziz, Affan, Azhari, Kausar, Farid yang telah memberikan dukungan dan segala bentuk bantuan yang telah diberikan.
7. Teman-teman Keluarga Besar Mahasiswa Teknik Komputer yang telah membagi ilmunya dalam akademik maupun dalam berorganisasi dan memberikan saran dalam perkuliahan.
8. Teman-teman tercinta program studi Teknik Komputer angkatan 2014 yang selalu memberikan semangat. Terima kasih atas semua doa dan dukungan baik dalam bentuk materil maupun non materil.
9. Dan orang-orang yang selalu mendukung dan mendoakan penulis yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa dalam laporan ini masih banyak kekurangan, oleh sebab itu penulis berharap adanya pengembangan lebih lanjut oleh pihak-pihak terkait. Semoga laporan ini dapat memberikan manfaat dan referensi untuk melakukan penelitian sebagai langkah penyempurnaan sistem.

Malang, 09 Mei 2018

Penulis
tdl.yus@gmail.com

ABSTRAK

Dunia robotika saat ini terus mengalami perkembangan. Di Indonesia sendiri pemerintah sangat mendukung dunia robotika dengan diadakannya berbagai macam kompetisi robotika yang salah satunya adalah Kontes Robot Indonesia. Dalam kontes tersebut terdapat berbagai macam robot yang dilombakan yang salah satunya adalah Kontes Robot Pemadam Api Indonesia dimana robot harus berjalan di arena pertandingan untuk menemukan dan memadamkan api. Sekarang ini banyak robot yang menggunakan konsep *hexapod* atau robot 6 kaki dengan *wall following* sebagai algoritma manuver. Tetapi dengan desain robot *hexapod* robot harus melakukan putaran sebesar 90 derajat pada saat ada halangan di depan sehingga banyak memakan waktu. Untuk itu diperlukan sebuah desain robot yang dapat melakukan manuver dengan baik dan cepat. Maka untuk mengatasi permasalahan tersebut, dilakukan pada penelitian ini implementasi algoritma *wall following* pada manuver robot *quadruped omni direction* dengan menggunakan metode fuzzy untuk pengambilan keputusan robot. Dengan desain robot *quadruped* dimungkinkan robot dapat melakukan manuver secara *omni direction* sehingga robot tidak perlu memutar dan didapatkan hasil yang lebih cepat. Dari hasil pengujian didapatkan robot yang melakukan manuver *omni direction* lebih cepat dibandingkan manuver robot memutar dengan selisih kecepatan rata-rata 4.1 detik. Robot dapat berjalan dan dapat melakukan manuver dengan baik menggunakan algoritma *wall following* dan logika *fuzzy* dengan tingkat keberhasilan mencapai 100%.

Kata kunci: robot *quadruped, wall following,* logika *fuzzy*

ABSTRACT

The world of robotics is now constantly developing. In Indonesia the government strongly supports the world of robotics with the holding of various kinds of robotics competition which one of them is Indonesia Robot Contest. In the contest there are various kinds of robots that are contested one of which is the Indonesian Fire Extinguisher Robot Contest where the robot must walk in the match arena to find and extinguish the fire. Today many robots use the hexapod or 6 foot robot concept with wall following as the maneuvering algorithm. But with robotic design robot hexapod must do 90 degrees rotation when there is a hitch in the front so need more time. For that required a robot design that can perform maneuvers properly and quickly. So to overcome these problems, in this research do the implementation of wall following algorithm on robot maneuver quadruped omni direction by using fuzzy method for robot decision making. With the design of robot quadruped robot is possible to maneuver in omni direction so that the robot does not need to rotate and get results faster. From the test results obtained robots who perform maneuvers omni direction is faster than the rotating robot maneuvering with the average speed difference of 4.1 seconds. The robot can run and can maneuver well using wall following algorithm and fuzzy logic with 100% success rate.

Keywords: *quadruped robot, wall following, fuzzy logic*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Robot adalah sebuah sistem mekanik yang mampu melakukan tindakan kompleks secara otomatis yang diprogram oleh komputer. Di Indonesia sendiri perkembangan dibidang robotika telah mengalami perkembangan yang sangat pesat. Salah satu ajang yang sebagai wadah pengembangan robot di bidang pendidikan adalah Kontes Robot Indonesia (KRI) yang diadakan setiap tahun oleh Kementrian Riset, Teknologi dan Pendidikan Tinggi Republik Indonesia. Ajang KRI terdiri dari 5 kategori, yaitu Kontes Robot Pemadam Api (KRPAI), Kontes Robot Sepak bola Humanoid (KRSBI Humanoid), Kontes Robot Sepak bola Beroda (KRSBI Beroda), Kontes Robot Abu Indonesia (KRAI) dan Kontes Robot Seni Tari Indonesia (KRSTI).

Pada Kontes Robot Pemadam Api (KRPAI), robot menggunakan kaki sebagai alat gerak robot. Terdapat aturan yang harus dipenuhi dan robot harus menyelesaikan tugas yang telah ditentukan. Terdapat sebuah arena yang telah ditentukan pada pertandingan. Arena pertandingan Kontes Robot Pemadam Api berbentuk persegi dengan didalamnya terdapat empat ruangan. Masing-masing ruangan berbeda ukuran dan posisi pintu dengan ruang lainnya. Selain keberhasilan robot dalam memadamkan api, pemenang pertandingan juga ditentukan oleh waktu yang dibutuhkan dalam menyelesaikan tantangan. Untuk itu diperlukan gerakan robot yang cepat dan efisien sehingga dapat menyelesaiakan tantangan dengan lebih cepat (KRI, 2017).

Pada saat ini terdapat dua jenis robot pemadam api berdasarkan jumlah kakinya. Yang pertama adalah *Hexapod*, yaitu robot yang mempunyai 6 kaki dan robot *Quadruped* adalah robot yang mempunyai 4 kaki. Desain robot *Quadruped* lebih memungkinkan robot untuk berjalan ke berbagai arah dengan bentuk gerakan yang sama sehingga robot tidak perlu melakukan putaran untuk berganti arah. Dengan desain seperti itu maka akan didapat sebuah manuver robot yang lebih cepat dan tidak membuang waktu untuk memutar posisi robot. Dengan menggunakan algoritma *wall following* sebagai navigasi utama robot menggunakan input dari sensor jarak, robot dapat mengidentifikasi dimana posisi dinding penghalang sehingga robot dapat menyesuaikan posisi terhadap dinding dan dapat menyusuri dinding tersebut. Algoritma *wall following* ini cocok digunakan pada lintasan KRPAI karena bentuk lintasan yang sudah diketahui dan robot harus menyusuri semua tempat agar mendapat bonus point. Untuk memberikan perhitungan yang lebih akurat terhadap pembacaan posisi robot dengan dinding, diperlukan sebuah metode yang mampu membuat kesimpulan yang akurat meskipun dibawah ketidak pastian yaitu menggunakan metode logika fuzzy. Dengan metode logika fuzzy akan didapatkan semua kondisi pada *rules* fuzzy sehingga robot akan mempunyai lebih banyak kesimpulan untuk terus bergerak sesuai dengan output yang telah diberikan berdasarkan aturan tersebut.

Berdasarkan fakta tersebut maka dibutuhkan sebuah robot yang dapat melakukan manuver tanpa memutar posisi arah robot dan robot dapat berjalan kesegala arah (*omni direction)* dengan menerapkan metode fuzzy sugeno pada algoritma *wall following* yang akan dikembangkan dalam skripsi ini.

1.2 Rumusan masalah

Dari penjelasan latar belakang tersebut, maka dapat dirumuskan suatu permasalahan sebagai berikut:

1. Bagaimana bentuk robot *quadruped* (berkaki empat) yang dapat melakukan manuver *omni direction*?
2. Bagaimana tingkat akurasi sensor jarak Srf08 yang digunakan pada robot?
3. Bagaimana penerapan metode fuzzy sugeno pada algoritma wall following robot quadruped?
4. Bagaimana hasil perbandingan kecepatan antara manuver robot *omni direction* dengan manuver robot memutar?

1.3 Tujuan

Tujuan yang ingin penulis capai pada penelitian ini adalah:

1. Membuat bentuk robot *quadruped* (berkaki empat) yang dapat melakukan manuver *omni direction*

2. Mengetahui tingkat akurasi sensor Srf08 yang akan digunakan pada robot

3. Menerapkan metode logika fuzzy model *sugeno* pada algoritma *wall following robot quadruped*

4. Mengetahui hasil perbandingan kecepatan antara manuver robot *omni direction* dengan manuver robot memutar

1.4 Manfaat

Manfaat yang diharapkan pada penelitian ini adalah dapat menjadi salah satu pemodelan robot yang dapat melakukan manuver dengan lebih cepat menggunakan metode fuzzy sugeno.

1.5 Batasan masalah

Batasan masalah dalam penelitian dan pembutuan sistem ditentukan agar permasalahan yang dirumuskan lebih fokus. Batasan pada penelitian ini adalah:

1. Robot dijalankan pada lintasan robot yang digunakan pada Kontes Robot Pemadam Api Indonesia

2. Robot menggunakan 4 kaki dengan 3 DOF (*Degree of Freedom*) pada masing-masing kaki

3. *Wall Following* yang digunakan adalah hanya pada bagian kanan robot

1.6 Sistematika Pembahasan

Penelitian skematik pembahasan dan penyusunan laporan penelitian implementatif masuk dalam *development* dapat diuraikan sebagai berikut.

BAB I Pendahuluan

Pada bab pendahuluan dijelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika penulisan.

BAB II Landasan Kepustakaan

Pada bab ini akan menjelaskan tentang landasan teori yang terkait dengan penelitian. Pada bab ini juga dijelaskan tentang penelitian serupa yang pernah dilakukan.

BAB III Metode Penelitian

Membahas tentang langkah kerja yang dilakukan dalam penelitian diantaranya studi literatur, analisis kebutuhan sistem dan perancangan sistem.

BAB IV Rekayasa Kebutuhan

Bab rekayasa kebutuhan menjelaskan terkait gambaran umum sistem, kebutuhan sistem meliputi kebutuhan fungsional, kebutuhan *non* fungsional kebutuhan perangkat lunak dan kebutuhan perangkat keras, kebutuhan komunikasi, batasan desain sistem.

BAB V Perancangan dan Implementasi Sistem

Bab ini membahas tentang perancangan sistem serta implementasi dari perancangan tersebut.

BAB VI Pengujian dan Analisis

Bab ini membahas tentang beberapa pengujian yang dilakukan yang kemudian dapat dianalisa.

BAB VII Kesimpulan dan Saran

Bab ini membahas kesimpulan yang diperoleh dari perancangan, implementasi dan pengujian sistem, serta saran-saran untuk pengembangan penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Kajian pustaka ini membahas penelitian yang sudah ada dan berkaitan dengan penelitian yang diusulkan. Pada penelitian ini, kajian pustaka diambil dari beberapa penelitian yang pernah dilakukan.

2.1.1 Pengembangan Logika *Fuzzy* pada Kendali Wall Following Mobile Robot

Dalam penelitian ini diusulkan kendali *mobile robot* menggunakan logika *fuzzy* karena mampu membuat kesimpulan bahkan dibawah ketidakpastian. Dengan menggunakan algortima *wall following* robot dapat mengidentifikasi dimana posisi dinding penghalang berada. Perilaku ini memungkinkan robot untuk mengikuti kontur objek seperti dinding dan rintangan di lingkungan dan juga dapat dikombinasikan dengan perilaku yang lebih cerdas untuk menyelesaikan tugas tingkat tinggi. Logika *fuzzy* menawarkan solusi yang menjanjikan untuk menangani ketidakpastian dunia nyata dengan meniru pengalaman manusia dalam bentuk aturan. Banyak peneliti telah menggunakan logika *fuzzy* dalam mengembangkan *controller* untuk perilaku *mobile robot* dengan algoritma *wall following*.

Pada penerapannya mobile robot tersebut menggunakan lima sensor ultrasonic dan menggunakan dua roda sebagai aktuator. Robot dikendalikan menggunakan *fuzzy incremental controller* (FIC) dan ditanamkan pada mikrokontroler PIC18F4550. FIC memandu robot untuk bergerak sepanjang dinding ke arah yang diinginkan dengan mempertahankan jarak konstan ke dinding. Dua sensor ultrasonik dipasang di sisi kiri robot untuk merasakan jarak dinding. Sinyal-sinyal dari sensor ini dikirim ke FIC yang kemudian digunakan untuk menentukan kontrol kecepatan dua motor DC. Gerakan robot diperoleh dengan membedakan kecepatan dua motor ini. Hasil eksperimen menunjukkan bahwa FIC berhasil mengendalikan robot untuk mengikuti dinding sebagai garis panduan dan memiliki kinerja yang baik dibandingkan dengan kontroler PID. Pengontrol yang diusulkan adalah dua input dan satu sistem output. Informasi jangkauan dari sensor ultrasonik digunakan sebagai bentuk input ke *controller*. Output dari pengendali logika *fuzzy* adalah nilai-nilai siklus dari sinyal PWM untuk mengendalikan kecepatan motor.

Pada kesimpulannya bahwa *controller fuzzy* yang dirancang menerima input dari kesalahan dan sensor kesalahan delta dan menghasilkan perintah motor sehingga robot mengikuti dinding dengan mempertahankan jarak yang ditetapkan dari dinding. Hasil eksperimen yang diperoleh setelah pengujian robot dalam lingkungan yang mengandung lintasan dinding yang berbeda telah menunjukkan validitas pengontrol yang diusulkan (Fahmizal, 2013).

2.1.2 Penerapan Metode *Fuzzy* Pada Robot Beroda Menggunakan Omni-Directional Wheels

Pada penelitian ini, penulis melakukan penerapan logika *fuzzy* sebagai algoritma sistem kendali robot yang diterapkan pada robot beroda menggunakan *omni-directional wheels*. Dengan menggunakan perpaduan antara kemampuan logika *fuzzy* dalam pengambilan keputusan set poin secara otomatis, diharapkan kontrol robot beroda ini lebih efektif dan stabil. Berdasarkan perancangan, pengujian, dan analisis yang telah dilakukan, maka dapat disimpulkan hal-hal yaitu dari hasil pengujian terhadap pembacaan sensor dan gerakan menggunakan roda *omni-directional wheels*, di dapatkan bahwa gerakan robot menjadi lebih bebas dan lebih leluasa dalam menelusuri ruangan.

Berdasarkan kesimpulan dari penelitian ini didapatkan bahwa robot yang menerapkan metode *fuzzy* dan menggunakan roda omni-directional bekerja lebih baik dari sebelumnya dengan persentase 82% dan 94%. Penggunaan roda omni-directional pada robot beroda membuat gerakan yang lebih bebas dari roda lain nya, sehingga sangat membantu robot dalam menelusuri dan menghindari semua rintangan. Penggunaan kontrol logika *fuzzy* membuat keputusan gerakan robot menjadi lebih banyak (Rahman, 2016).

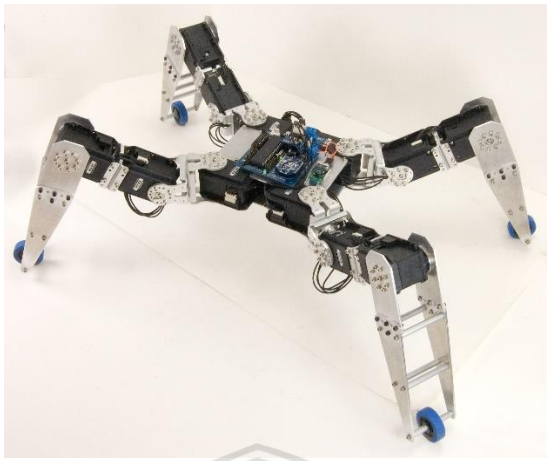
2.2 Dasar Teori

Dasar teori membahas berbagai teori yang diperlukan dalam menyusun penelitian yang diusulkan.

2.2.1 Robot Quadruped

Robot merupakan perangkat otomatis yang mampu bergerak sendiri untuk menyelesaikan sebuah pekerjaan. Banyak sekali jenis-jenis robot yang ada. Namun berdasarkan alat geraknya robot diklasifikasikan menjadi 2 (dua) jenis yaitu robot beroda dan robot berkaki. Robot beroda adalah robot yang mampu bermanuver dengan menggunakan roda, baik dengan dua roda atau lebih dari dua roda. Robot berkaki adalah robot yang bermanuver dengan kaki-kaki buatan, baik dengan 2 kaki yang sering disebut dengan robot humanoid, berkaki tiga (tripod), berkaki empat (quadrapod), robot berkaki enam(hexapod) dan robot berkaki banyak lainnya. Pergerakan robot dengan menggunakan roda hampir tidak mengalami masalah dalam pengaturannya kecuali untuk robot beroda dua.

Robot *quadruped* atau robot berkaki 4 adalah robot yang lebih adaptif terhadap medan tempuh dibandingkan robot beroda dalam kasus penggunannya pada kegiatan eksplorasi daratan. Walaupun secara fungsional robot berkaki lebih unggul, robot berakaki memilki permasalahannya sendiri, yaitu kontrol gerak yang lebih kompleks dibandingkan robot beroda, maka dari itu kita harus dapat memilih metode yang tepat untuk digunakan pada robot. Bentuk robot *quadruped* dapat dilihat pada Gambar2.1 (Tony, 2012).



Gambar 2.1 *Quadruped Robot*

Sumber: (Robotshop, 2010)

2.2.2 Logika *Fuzzy*

Logika *fuzzy* adalah peningkatan dari logika Boolean yang mengenalkan konsep kebenaran sebagian. Pada logika klasik Boolean diyatakan bahwa segala hal dapat diekspresikan dalam istilah binary (0 atau 1, hitam atau putih, ya atau tidak). Sementara itu logika *fuzzy* menggantikan kebenaran boolean dengan tingkat kebenaran. Logika fuzzy diperkenalkan oleh Dr. Lotfi Zadeh dari Universitas California, Berkeley pada 1965.

Menurut Budiharto (2014), logika *fuzzy* memiliki banyak kelebihan, dan alasan mengapa menggunakan logika *fuzzy* diantaranya adalah konsep logika *fuzzy* mudah dipahami karena konsep matematis yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti, sangat fleksibel, memiliki toleransi terhadap data-data yang tidak tepat, mampu memodelkan fungsi linear yang sangat kompleks, dapat membangun dan mengaplikasikan pelatihan, dapat bekerjasama dengan teknik-teknik kendali secara konvensional dan didasarkan pada bahasa alami.

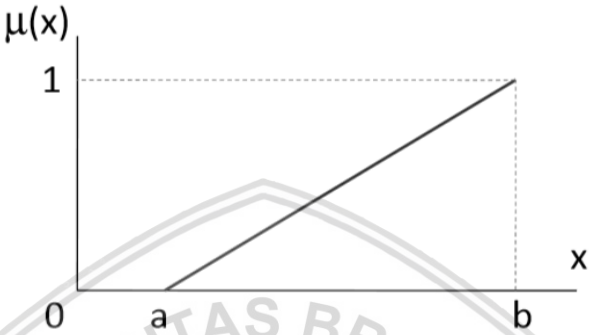
Dengan adanya sistem kontrol seperti logika *fuzzy* ini, perilaku robot akan semakin dinamis dengan adanya istilah tingkat keabuan dalam logika *fuzzy* itu sendiri. Robot tidak lagi hanya mengenal istilah 0 dan 1 atau hitam dan putih, robot akan terlihat lebih cerdas dalam berbagai kondisi yang akan dibuat (Budiharto W, 2014).

2.2.2.1 Logika Fuzzy Sugeno

Fuzzy metode Sugeno diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985. Terdapat dua model pada sistem inferensi *fuzzy* ini yaitu model TSK orde-0 dan model TSK orde-1. Terdapat tiga tahapan dalam proses logika *fuzzy* yaitu fuzzifikasi, inferensi dan yang terakhir adalah defuzzifikasi.

1. Fuzzyfikasi

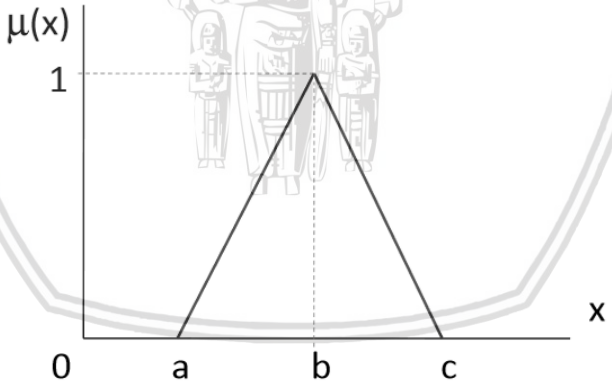
Fuzzifikasi madalah proses memetakan nilai kedalam himpunan *fuzzy* input dan menentukan derajat keanggotaan himpunan *fuzzy*. Terdapat beberapa fungsi derajat keanggotaan himpunan *fuzzy* antara lain fungsi linier, fungsi segitiga dan fungsi trapezium. Fungsi-fungsi derajat keanggotaan *fuzzy* tersebut dapat dilihat pada Gambar 2.2, Gambar 2.3 dan Gambar 2.4.



$$\mu(x) = \begin{cases} 0; & x \le a \\ (x-a)/(b-a); & a \le x \le b \\ 1; & x \ge b \end{cases}$$

Gambar 2.2 Grafik Fungsi Keanggotaan Linear

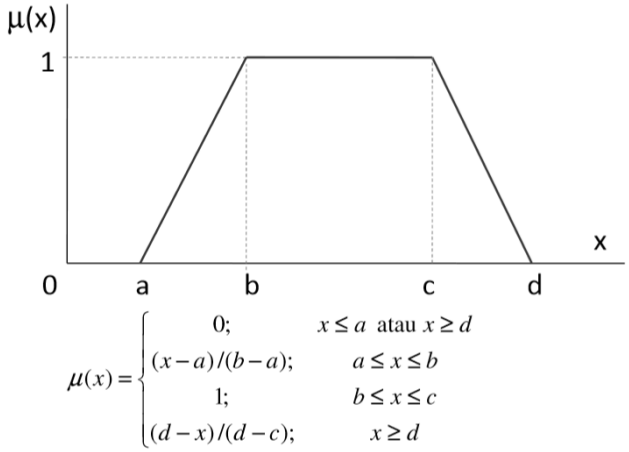
Sumber: (Kusumadewi, 2013)



$$\mu(x) = \begin{cases} 0; & x \le a \text{ atau } x \ge c \\ (x-a)/(b-a); & a \le x \le b \\ (b-x)/c-b; & b \le x \le c \end{cases}$$

Gambar 2.3 Grafik Fungsi Keanggotaan Segitiga

Sumber: (Kusumadewi, 2013)



Gambar 2.4 Grafik Fungsi Keanggotaan Trapesium

Sumber: (Kusumadewi, 2013)

2. Inferensi

Adalah fungsi menyusun basis aturan, yaitu aturan-aturan (*rule*) berupa implikasi *fuzzy* yang menyatakan relasi antar variable input dengan output. Fungsi inferensi yang digunakan adalah metode Min (minimum) yaitu solusi himpunan *fuzzy* diperoleh dengan cara mengambik nilai terkecil aturan. Kemudian apabila sistem terdiri dari beberapa *rule* maka inferensi diperoleh menggunakan metode Max (maksimum) yaitu solusi himpunan *fuzzy* yang diperokeh dengan cara mengambil nilai terbesar aturan.

3. Defuzzifikasi

Defuzzifikasi adalah tahap merubah output *fuzzy* menjadi crisp value berdasarkan fungsi keanggotaan himpunan *fuzzy*. Nilai defuzzifikasi berasal dari komposisi aturan. Hasil output yang dihasilan merupakan bilangan real yang tegas. Sehingga apabila diberikan suatu himpunan *fuzzy* dengan rentang tertentu, maka dapat diambil suatu nilai tegas sebagai output.

2.2.3 Algoritma Wall Following

Wall following merupakan salah satu metode navigasi robot yang pada dasarnya algoritma ini bertujuan untuk menjaga agar jarak robot pada dinding tetap pada batas yang diinginkan sementara robot terus bergerak maju. Terdapat beberapa situasi dimana algoritma ini dapat digunakan sebagai pemecahan permasalahan diantaranya yang pertama adalah ketika sensor tidak dapat memberikan pengetahuan tentang bentuk atau ukuran rintangan, maka diperlukan robot untuk mengikuti kontur dinding penghalang tersebut. Yang kedua adalah ketika *dead-reckoning* tidak dapat dilakukan maka diperlukan robot untuk menyusuri dinding sampai pada tujuan yang diinginkan. Salah satu keuntungannya adalah tidak perlu adanya garis penuntun ataupun suatu tanda khusus sebagai arahan bagi robot. Cara kerjanya adalah dengan mengatur jarak dinding dengan robot tetap konstan. Bila terjadi perubahan, maka robot akan

bergerak untuk kemudian menyesuaikan jarak lagi. Proses ini akan dilakukan secara berulang-ulang (Wang, 2011). Terdapat 4 metode pengukuran jarak robot dengan dinding yaitu:

1. ***Contact***

Robot menggunakan saklar mekanik yang merasakan sentuhan dengan dinding. Ini adalah metode yang paling mudah namun saklar akan cenderung mengalami kerusakan mekanis setelah beberapa waktu.

2. ***Noncontact Active Sensor***

Robot menggunakan sensor aktif yang beroperasi dalam jarak dekat seperti infra merah atau ultrasonik untuk mengukur jarak antara dinding dengan robot.

3. ***Noncontact Pasive Sensor***

Robot memakai sensor pasif seperti saklar Hall effect untuk mengukur jarak antara robot dengan dinding. Pada kasus ini, dinding harus berbahan logam atau dipasangai kabel elektrik agar sensor dapat menangkap medan magnetik saat robot mendekati dinding.

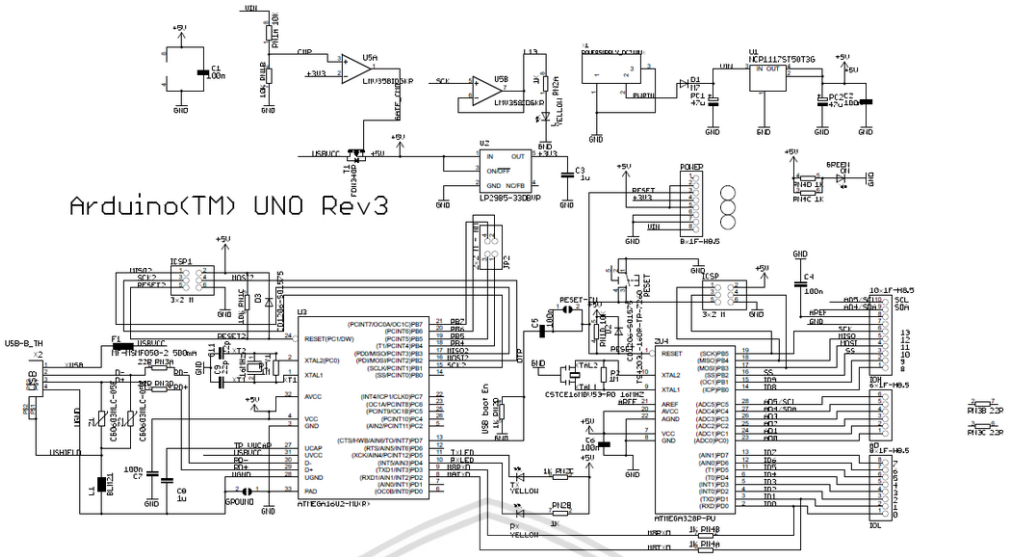
4. ***Soft Contact***

Robot menggunakan bahan mekanik untuk mendeteksi sentuhan dengan dinding, namun sentuhan ini diperhalus dengan memasang material lunak atau lentur contohnya roda dari busa atau karet. Kelebihan dari metode ini adalah berkurangnya kerusakan mekanis.

2.2.4 Arduino Uno

Arduino Uno merupakan mikrokontroler berbasis ATmega328. Arduino uno mempunyai 14 pin input / output digital dengan 6 pin yang dapat digunakan sebagai PWM (Pulse Width Modulation), 6 input analog, quartz crystal 16 MHz, koneksi USB, sebuah sambungan power, header ICSP dan tombol reset. Semua hal tersebut diperlukan untuk mendukung mikrokontroler, pengguna hanya perlu menghubungkan Arduino/ Genuino Uno ke komputer dengan kabel USB dan memberikan power dengan adaptor AC-DC atau baterai untuk memulai (Arduino2016).

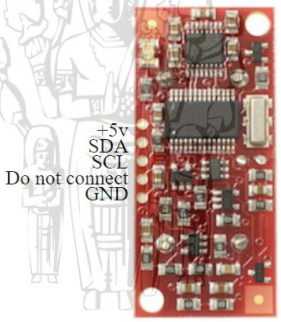
Arduino Uno dapat diprogram menggunakan Arduino Software (IDE) dengan memilih board pada menu *tools* sesuai dengan board yang akan digunakan. ATmega328 di Arduino Uno diprogram ulang dengan bootloader yang memungkinkan untuk mengunggah kode baru tanpa menggunakan programmer perangkat keras eksternal. Adapun skematik dari Arduino Uno dapat dilihat pada Gambar 2.5.



Gambar 2.5 Skematik Arduino Uno

Sumber: (Arduino, 2016)

2.2.5 Sensor SRF08



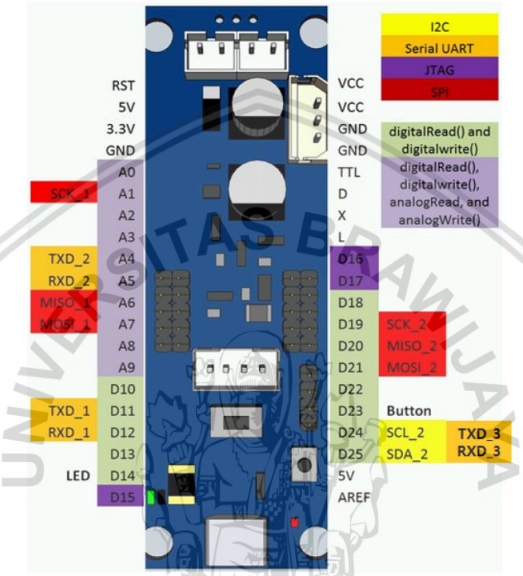
Gambar 2.6 Pin Komunikasi SRF08

Sumber: (Robot-Elektronik, 2017)

Sensor SRF08 adalah sensor ultrasonic untuk mengetahui jarak. Komunikasi SRF08 dengan mikrokontroller adalah melalui jalur I2C dengan menghubungkan pin SDA dan SCL yang dapat dilihat pada Gambar 2.6. Komunikasi ini tersedia pada pengendali populer seperti OOPic dan Stamp BS2p, serta berbagai macam mikro kontroler. Alamat pengiriman default dari SRF08 adalah 0xE0. Hal ini dapat diubah oleh pengguna ke 16 alamat yang tersedia, yaitu E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC atau FE, sehingga sampai 16 sensor sekaligus dapat digunakan. Selain alamat tersebut, semua sonar di bus I2C akan merespons alamat 0 - alamat General Broadcast. Ini berarti bahwa menulis perintah mulai ke alamat I2C 0 (0x00) akan memulai semua sonar mulai pada waktu bersamaan. Hasilnya harus dibaca secara terpisah dari setiap alamat sebenarnya dari sonar (Robot-Elektronik, 2017).

2.2.6 OpenCM9.04

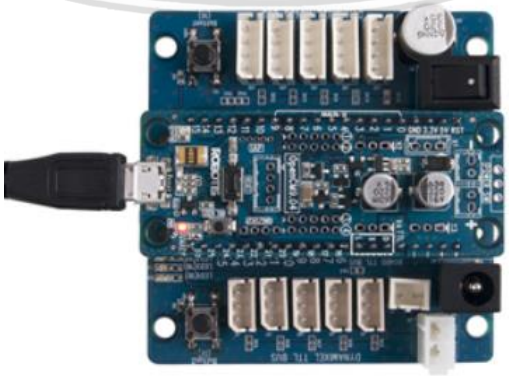
OpenCM9.04 adalah papan mikrokontroler yang berbasis pada 32bit ARM Cortex-M3. Skema OpenCM9.04 dan kode sumber bersifat open-source. Tresedia 3 tipe dari OpenCM9.04 yaitu tipe A, tipe B dan tipe C. Perbedaan antara tipe A, tipe B dan tipe C adalah ketersediaan konektor. OpenCM9.04 mempunyai kelebihan diantaranya sudah terdapat pin analog didalamnya. Dengan mikrokontroller tersebut juga dapat langsung digunakan untuk mengontrol servo dinamiksel AX-12. Konfigurasi pin GPIO OpenCM9.04 dapat dilihat pada Gambar 2.7 (ROBOTIS, 2010).



Gambar 2.7 Pin GPIO OpenCM9.04

Sumber: (ROBOTIS, 2010)

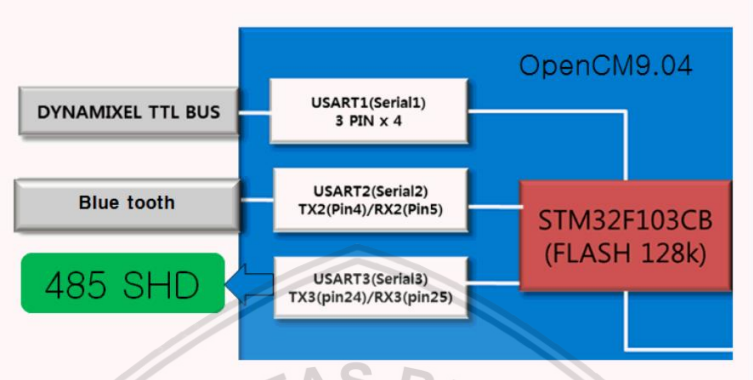
2.2.7 Board Ekpansi OpenCM 485



Gambar 2.8 layout OpenCM 485 dengan OpenCM9.04

Sumber: (ROBOTIS, 2010)

Gambar 2.8 adalah gambar *board expantion* OpenCM 485 yang merupakan ekspansi untuk mengontrol servo dinamiksel. Ekspansi OpenCM 485 terhubung ke pengendali OpenCM9.04. Papan pengontrol OpenCM9.04 diperlukan untuk menggunakan OpenCM 485 Expansion Board. Kontroler OpenCM9.04 mendukung Dynamixels menggunakan RS-485 dan TTL. Terdapat beberapa jalur komunikasi yang dapat digunakan seperti pada gambar 2.9.

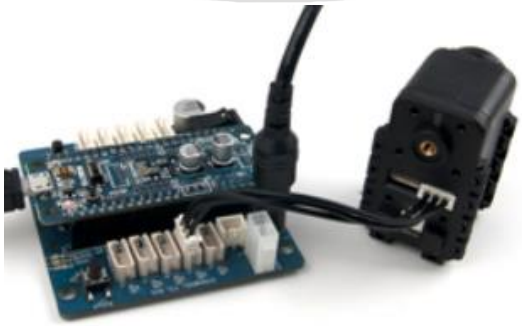


Gambar 2.9 Jalur Komunikasi OpenCM9.04 dengan OpenCM 485

Sumber: (ROBOTIS, 2010)

2.2.8 Servo Dinamixel AX-12A

Aktuator servo AX-12A dari Robotis adalah aktuator paling maju yang ada di pasaran saat ini dan telah menjadi standar defacto untuk generasi berikutnya robotika. Servo AX-12A memiliki kemampuan untuk melacak kecepatan, suhu, posisi poros, tegangan, dan beban. Seolah-olah ini tidak cukup, algoritma kontrol yang digunakan untuk mempertahankan posisi poros pada aktuator ax-12 dapat disesuaikan secara individual untuk masing-masing servo, yang memungkinkan mengendalikan kecepatan dan kekuatan respons motor. Semua manajemen sensor dan kontrol posisi ditangani oleh mikrokontroler servo built-in. Pendekatan terdistribusi ini membuat pengendali utama bebas melakukan fungsi lainnya. Adapaun layout komunikasi servo AX-12 dengan OpenCM dapat dilihat pada Gambar 2.10 (ROBOTIS, 2006).



Gambar 2.10 Layout Komunikasi Servo AX-12 dengan OpenCM

Sumber: (ROBOTIS, 2010)

2.2.9 Pemrograman Arduino

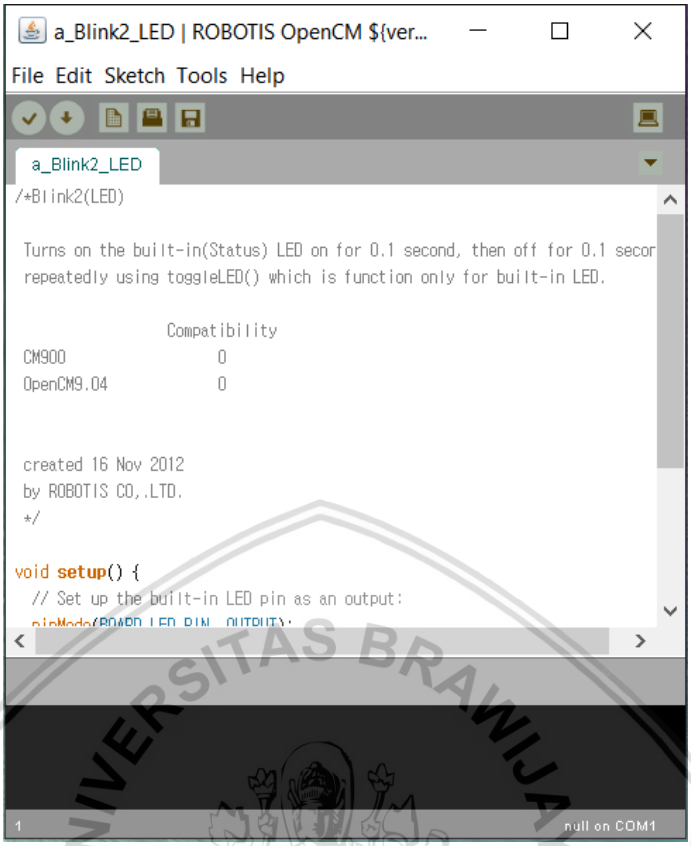
Arduino mempunyai software IDE Arduino untuk menuliskan kode program serta meng-*upload* kode program tersebut. Software tersebut dapa digunakan di sistem operasi Windows, Linux dan Mac OS. Pemrograman ini adalah pemrograman tingkat rendah, yang berarti pemrograman dilakukan langsung ke hardware. Layout IDE arduino dapat dilihat pada Gambar 2.11 (Arduino, 2016).

Gambar 2.11 Software IDE Arduino

Sumber: (Arduino, 2016)

2.2.10 Pemrograman OpenCM9.04

OpenCM9.04 merupakan salah satu produk ROBOTIS berupa *servo controller open source*. Hal ini berarti pengguna dapat menggerakkan servo *Dynamixel* menggunakan kode program sendiri. Untuk dapat memprogram OpenCM9.04 dibutuhkan software IDE berupa ROBOTIS OpenCM. Sumber ROBOTIS OpenCM dilepaskan di bawah lisensi masing-masing pengarangnya. Software IDE ini mirip dengan software IDE Arduino. Layout IDE Opencm dapat seperti pada Gambar 2.12 (ROBOTIS, 2010).



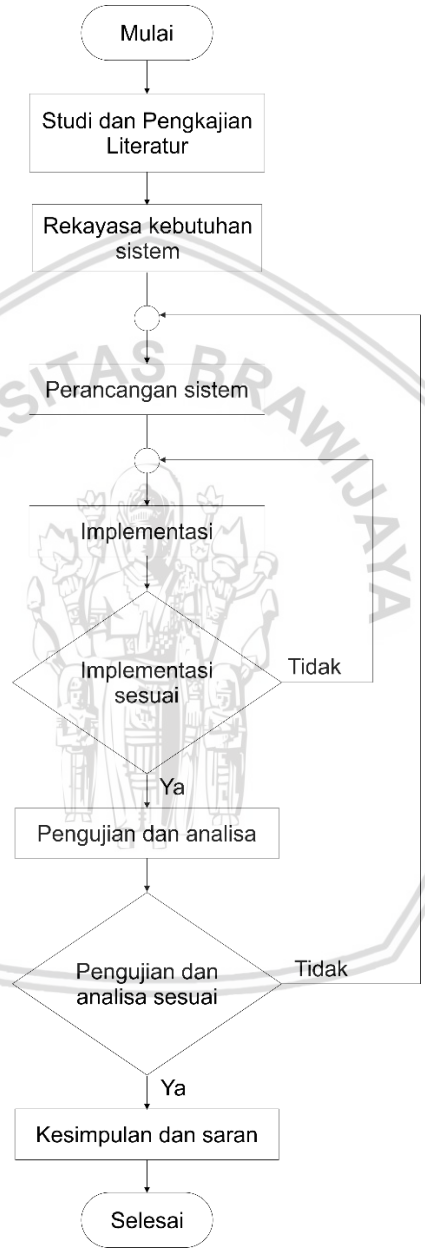
Gambar 2.12 Software IDE OpenCM9.04

Sumber: (Robotis, 2010)

BAB 3 METODOLOGI

3.1 Alur Metodologi Penelitian

Dari Alur metodologi penelitian yang dilakukan secara umum dapat dilihat dari diagram alir pada Gambar 3.1.



Gambar 3.1 Alur Metodologi Penelitian

Berdasarkan gambar alur metodologi penelitian pada Gambar 3.1, semua proses dilakukan secara terurut dimulai dari studi dan pengkajian literatur sampai dengan kesimpulan. Beberapa tahapan mempunyai syarat agar dapat dilanjutkan

ke tahapan berikutnya, seperti jika implementasi sesuai dengan semua yang sudah dirancang sebelumnya pada implementasi sistem, maka sistem dinyatakan siap untuk dilakukan pengujian dan analisis. Sebaliknya jika implementasi tidak sesuai dengan yang dirancang maka akan dilakukan pengulangan terhadap implementasi sistem. Implementasi sesuai atau tidak berdasarkan dari hasil sistem ketika dijalankan. Ketika pada tahap dijalankan sudah sesuai yang dirancang, maka implementasi berhasil. Begitu juga dengan pengujian dan analisis, ketika pengujian dan analisis sesuai dengan hipotesis awal, maka dapat ditarik kesimpulan dari keseluruhan tahapan. Jika tidak sesuai, maka akan kembali ke tahapan perancangan sistem.

3.1.1 Studi Literatur

Studi literatur yang dilakukan bertujuan untuk mengkaji hal-hal yang berhubungan dengan teori-teori yang mendukung dalam perencanaan alat. Adapun teori-teori yang dikaji adalah sebagai berikut:

1. Konstruksi robot pemadam api berkaki 4 (*quadruped*) dengan 3 DOF dimasing-masing kaki.
2. Pengkajian pembacaan sensor jarak SRF08 pada mikrokontroller Arduino uno.
3. Kontrol servo dinamiksel AX-12 pada openCM 9.04 dan ekspansi openCM 485
4. Pengendalian kecepatan servo menggunakan logika fuzzy sugeno

3.1.2 Rekayasa Kebutuhan Sistem

Rekayasa kebutuhan system meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Perangkat lunak yang dibutuhkan dalam membangun sistem ini adalah perangkat lunak yang dapat digunakan untuk membuat program pada sistem meliputi:

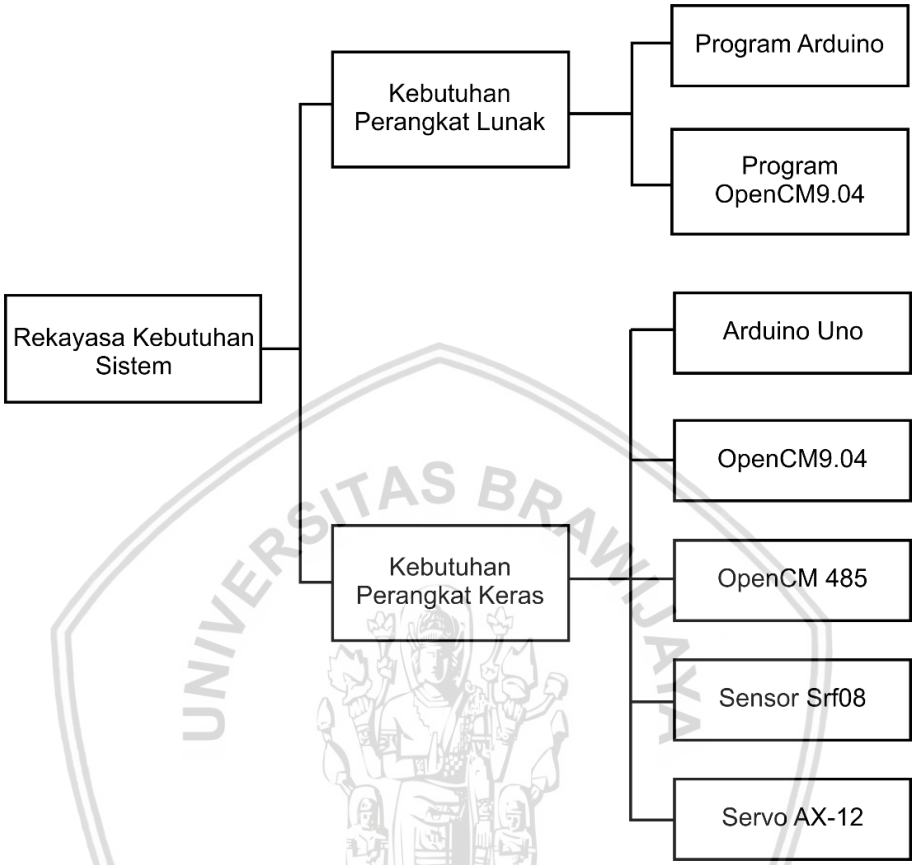
1. Perangkat lunak yang dapat digunakan untuk membuat program dan dimasukkan ke Mikrokontroller yang digunakan.

2. Perangkat lunak yang dapat digunakan untuk membuat program menggerakkan penggerak servo.

Sedangkan perangkat keras yang dibutuhkan dalam membangun sistem ini mempunyai spesifikasi sebagai berikut:

1. Perangkat keras yang dapat di-*upload* suatu program untuk membaca nilai sensor yang masuk kemudian mengirimkan data tersebut secara serial ke komputer untuk dilakukan proses perhitungan logika *fuzzy*.
2. Perangkat keras yang dapat dibaca kemudian menjadi input Mikrokontroler yang digunakan.
3. Perangkat keras yang dapat mengkontrol alat penggerak berupa servo dinamixel AX-12.

Analisis kebutuhan sistem dibagi menjadi dua meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Rekayasa kebutuhan digambarkan pada pohon rekayasa kebutuhan sistem pada Gambar 3.2.



Gambar 3.2 Pohon Rekayasa Kebutuhan Sistem

3.1.2.1 Kebutuhan Perangkat Lunak

Pada kebutuhan perangkat lunak, penelitian ini membutuhkan sebuah IDE yang dapat meng-compile dan meng-*upload* kode program yang sudah dikerjakan untuk membaca nilai sensor ke dalam mikrokontroler. Sistem ini menggunakan aplikasi IDE Arduino untuk men-*download* program ke Arduino dan aplikasi Robotis OpenCM untuk mendownload program ke openCM 9.04. Bahasa yang digunakan oleh IDE ini menyesuaikan dari jenis mikrokontroler yang digunakan, yaitu menggunakan bahasa C.

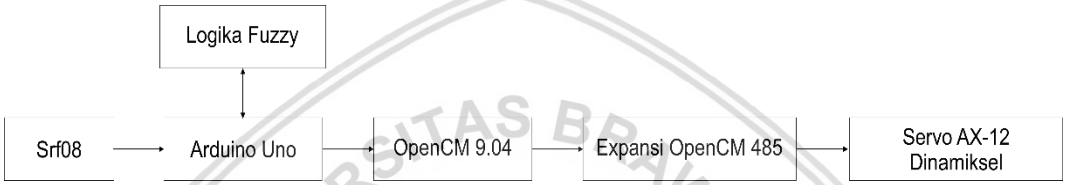
3.1.2.2 Kebutuhan Perangkat Keras

Pada penelitian ini, peneliti membuat sebuah sistem robot yang dapat berjalan menyusuri sebuah arena dengan pembacaan dari sebuah sensor jarak. Kebutuhan perangkat keras membutuhkan Mikrokontroller sebagai pemroses data dari sebuah sensor dan sebuah *servo controller* sebagai penggerak *actuator.* Kedua perangkat keras tersebut harus dapat berkomunikasi untuk pengiriman data. Yang dalam penelitian ini peneliti menggunakan Arduino Uno sebagai

Mikrokontroller dan sebagai penggerak servo menggunakan OpenCM9.04 dan ekspansi OpenCM 485. Inputan sistem menggunakan sensor SRF08 yaitu berupa sensor jarak yang menerapkan prinsip gelombang ultrasonik. Sedangkan untuk *akuator* adalah berupa servo dinamixel AX-12 yang akan dirangkai menjadi sebuah kaki robot.

3.2 Perancangan Sistem

Perancangan sistem dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan yang meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Setelah kebutuhan sistem terpenuhi selanjutnya adalah perancangan perangkat lunak dan perancangan perangkat keras yang tergambar pada blog diagram pada gambar 3.3.



Gambar 3.3 Blok Diagram Sistem

Sensor Srf08 akan membaca jarak yang akan menjadi inputan dari sistem. Dari inputan tersebut akan diproses pada mikrokontroller arduino uno menggunakan logika *fuzzy* model sugeno. Setelah perhitungan selesai maka output dari sistem akan menggerakan servo yang difungsikan sebagai kaki oleh servo kontroller openCM 904 dan ekspansi openCM 485 yang berkomunikasi dengan Arduino dengan cara serial.

3.3 Implementasi Sistem

Implementasi sistem meliputi proses perancangan sistem sampai dengan pada hasil akhir penelitian. Berikut adalah tahapan implementasi pada sistem:

1. Implementasi servo *dynamixel* AX-12 dalam pembuatan *body* kaki robot *quadruped.*
2. Implementasi sensor Srf08 dalam pembuatan *layer body* sensor.
3. Implementasi OpenCM9.04 dan OpenCM 485 dalam pembuatan gerakan (*motion*) robot *quadruped*.
4. Komunikasi antara Mikrokontroller Arduino Uno dengan Servo Kontroller OpenCM9.04 dan OpenCM 585 menggunakan komunikasi data serial UART.
5. Implementasi pembacaan sensor Srf08 menggunakan Mikrokontroller Arduino Uno.
6. Implementasi metode *fuzzy* pada algoritma *Wall Following*.

3.4 Pengujian Sistem

Tedapat beberapa rancangan scenario pengujian yang akan dilakukan antara lain:

1. Pengujian sensor Srf08 dengan membandingkan nilai yang terbaca pada Arduino dengan nilai sesungghnya.
2. Pengujian dilakukan dalam lintasan sejauh 224cm dengan ditengah-tengah terdapat masinig-masing 1 belokan kekanan dan kekiri.
3. Pengujian dilakukan dengan menggeser nilai derajat keanggotaan input *fuzzy* sensor denpan, sensor kanan dan sensor miring dengan masing-masing 3 derajat keanggotaan yang berbeda.
4. Pengujian dilakukan dengan mmembandingkan antara robot yang melakukan manuver tanpa memutar badan dengan robot yang melakukan menuver dengan memutar badan robot dengan jumlah pengujian sebanyak 10 kali percobaan.

3.5 Analisis dan Pembahasan

Setelah semua tahap dilakukan, mulai dari implementasi dan pengujian, dilakukan tahap selanjutnya, yaitu analisis dan pembahasan. Pada tahap ini dilakukan analisis terhadap hasil semua hasil pengujian dan melakukan pembahasan, untuk menentukan nilai dari hasil pengujian sudah benar atau tidak, sehingga dapat dikatakan pengujian terhadap sistem berhasil.

3.6 Kesimpulan

Setelah proses perancangan, implementasi sistem, pengujian, analasis serta pembahasan dilakukan, maka dilakukan penarikan kesimpulan. Kesimpulan berisi gambaran dari hasil iemplementasi menggunakan metode *fuzzy.* Kesimpulan juga merupakan jawaban dari rumusan masalah yang telah ditetapkan diawal penelitian.

BAB 4 REKAYASA KEBUTUHAN

Bab rekayasa kebutuhan menjelaskan terkait gambaran umum sistem, kebutuhan sistem meliputi kebutuhan fungsional, kebutuhan *non* fungsional kebutuhan perangkat lunak dan kebutuhan perangkat keras, kebutuhan komunikasi, batasan desain sistem serta alur kerja dari sistem.

4.1 Gambaran Umum Sistem

Robot *quadruped* merupakan robot yang memiliki 4 kaki. Dibutuhkan sebuah manuver agar robot tersebut dapat berjalan dengan lebih cepat. Dengan robot *quadruped* ini memungkinkan robot tersebut bermanuver tanpa harus memutar badan robot atau berbelok arah. Dengan demikian manuver robot akan lebih efisien dan robot dapat berjalan dengan lebih cepat.

4.2 Kebutuhan Sistem

Pada bagian kebutuhan sistem dijelaskan kebutuhan fungsional, kebutuhan non fungsional, kebutuhan perangkat keras dan kebutuhan perangkat lunak, sehingga diharapkan akan menjadi lebih mudah dalam melakukan desain dan implementasi sistem.

4.2.1 Kebutuhan Perangkat Keras

Pada penelitian ini dibutuhkan perangkat keras yang digunakan antara lain:

1. **Mikrokontroller Arduino Uno**

   Mikrokontroller Arduino adalah sebagai konroller utama yang dapat menerima inputan sensor, melakukan pengolahan metode dan algoritma dan mengirimkan perintah bagaimana robot harus bergerak. Adapun tabel spesifikasi Arduino Uno dapat dilihat pada Tabel 4.1

Tabel 4.1 Spesifikasi Arduino Uno

Microcontroller	*ATmega328P*
Operating Voltage	*5V*
Input Voltage (recommended)	*7-12V*
Input Voltage (limit)	*6-20V*
Digital I/O Pins	*14 (of which 6 provide PWM output)*
PWM Digital I/O Pins	*6*
Analog Input Pins	*6*
DC Current per I/O Pin	*20 mA*
DC Current for 3.3V Pin	*50 mA*
Flash Memory	*32 KB (ATmega328P) of which 0.5 KB used by bootloader*
SRAM	*2 KB (ATmega328P)*
EEPROM	*1 KB (ATmega328P)*

Clock Speed	*16 MHz*
LED_BUILTIN	*13*
Length	*68.6 mm*
Width	*53.4 mm*
Weight	*25 g*

Sumber: (Arduino, 2016)

2. Sensor Srf08

Sensor Srf08 berfungsi untuk mendeteksi posisi robot dari dinding penghalang. Niai yang dihasilkan berupa nilai jarak dengan satuan centimeter. Tabel spesifikasi sensor Srf08 dapat dilihat pada Tabel 4.2

Tabel 4.2 Spesifikasi Sensor Srf08

Voltage	*5v*
Current	*15MA Typ. 3mA Standby*
Frequency	*40KHz*
Maximum Range	*6m*
Minimum Range	*3cm*
Connection	*Standard IIC Bus*
Size	*43 mm w x 20mm d x 17mm h*
Weight	*0.4 oz*

Sumber: (Robot-Elektronik, 2017)

3. OpenCM9.04 dan Board Ekspansi OpenCM 485

OpenCM9.04 dan board ekspansi OpenCM 485 berfungsi untuk menggerakkan actuator berupa servo dynamiksel AX-12. OpenCM 9.04 memiliki IDE untuk melakukan programing gerakan robot. Tabel spesifikasi OpenCM9.04 dan OpenCM485 dapat dilihat pada Tabel 4.3 dan Tabel 4.4.

Tabel 4.3 Spesifikasi OpenCM9.04

CPU	*STM32F103CB (ARM Cortex-M3)*
Voltage	*5V-16V*
I/O	*GPIO 26*
Timer	*4 (16bit)*
Flash	*128 Kbytes*
SRAM	*20 Kbytes*
Clock	*72Mhz*

USB	*1 (2.0 Full-Speed) Micro B Type*
USART	*3*
SPI	*2*
I2C (TIWI)	*2*
Debug	*JTAG & SWD*
Dynamixel TT: Bs 3 pin	*4*
Size	*27mm x 66.5mm*

Sumber: (ROBOTIS, 2010)

Tabel 4.4 Spesifikasi OpenCM 485

Description	*Specification*
Size	*68 mm X 66.5 mm x 16 mm*
Weight	*32 g*
Input Voltage	*5~30V*
Power	*SMPS, LIPO, DXL PRO 24V*
Power Switch	*1*
DYNAMIXEL Port	*4Pin x 5, 3Pin x 5*
Buttons	*2*
LED	*5*

Sumber: (ROBOTIS, 2017)

4. Servo AX-12

Robot quadruped adalah robot yang mempunyai 4 kaki yang menggunakan servo dxnamixel AX-12 dengan lebar tidak melebihi 30cm dan tinggi robot tidak lebih dari 27cm pada saat posisi robot berdiri. Adapun spesifikasi dari sero dinamiksel AX-12 dapat diliihat pada Tabel 4.5.

Tabel 4.5 Spesifikasi Servo Dynamixel AX-12

Description	*Specification*
Weight	*54.6g*
Dimension	*32mm x 50mm x 40mm*
Gear Ratio	*254 : 1*
Operation Voltage (V)	*12*
Stall Torque (N.m)	*1.5 (12V)*
Stall Current (A)	*1.5*
No Load Speed (RPM)	*59 (12V)*
Motor	*Cored Motor*
Minimum Control Angle	*about 0.29 degrees x 1,024*

Operating Range	*Actuator Mode : 300 degrees Wheel Mode : Endless turn*
Operating Voltage	*9~12V (Recommended voltage : 11.1V)*
Max. Current	*900mA*
Standby Current	*50mA*
Operating Temperature	*-5°C ~ 70°C*
Command Signal	*Digital Packet*
Protocol	*Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)*
Link (physical)	*TTL Level Multi Drop (daisy chain type Connector)*
ID	*254 ID (0~253)*
Baud Rate	*7843bps ~ 1 Mbps*
Feedback Functions	*Position, Temperature, Load, Input Voltage, etc.*
Material	*Case : Engineering Plastic Gear : Engineering Plastic*
Position Sensor	*Potentiometer*
Default Setting	*ID #1 (1 Mbps)*

Sumber: (ROBOTIS, 2006)

5. **Modul Regulator Stepdwon LM2596**

   Modul regulator stepdown digunakan untuk menurunkan tegangan dari baterai Li-Po sebesar 12v menjadi tegangan sebesar 5v.

4.2.2 Kebutuhan Perangkat Lunak

Pada penelitian ini dibutuhkan beberapa perangkat lunak antara lain:

1. **IDE Arduino**

   Perangkat lunak ini digunakan untuk menuliskan program dan meng-*upload* program tersebut ke Mikrokontroller Arduino Uno.

2. **IDE Robotis OpenCM**

   Perangkat lunak adalah produk dari Robotis yang berfungsi untuk menuliskan kode program dan meng-*upload* program tersebut ke servo controller OpenCM9.04.

4.3 Batasan Desain Sistem

Agar sistem ini dapat dilakukan sesuai dengan harapan, maka perlu diterapkan batasan-batasan implementasi desain sistem, antara lain:

1. Sistem menggunakan mikrokontroller Arduino Uno dan servi controller OpenCM9.04

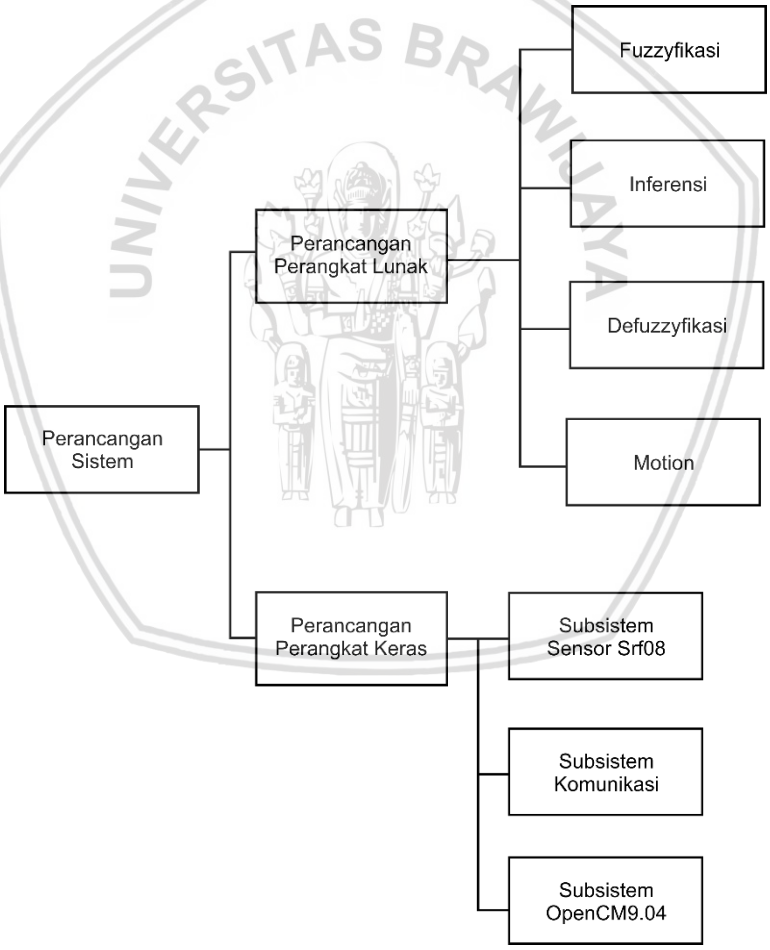
2. Sistem menggunakan sensor Srf08
3. Sensor Srf08 berjumlah 12 buah sensor
4. Robot yang digunakan adalah robot quadruped dengan tinggi maksimal 27cm menggunakan servo AX-12.
5. Robot bergerak ketika data dari arduino dikirimkan ke OpenCM9.04 untuk menggerakkan robot.
6. Robot bergerak disebuah lapangan berbentuk labirin yang mempunyai 4 buah ruangan.
7. Pengiriman data serial dari mikrokontroller Arduino ke OpenCM9.04 tidak dijelaskan secara rinci pada implementasi.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

5.1 Perancangan Sistem

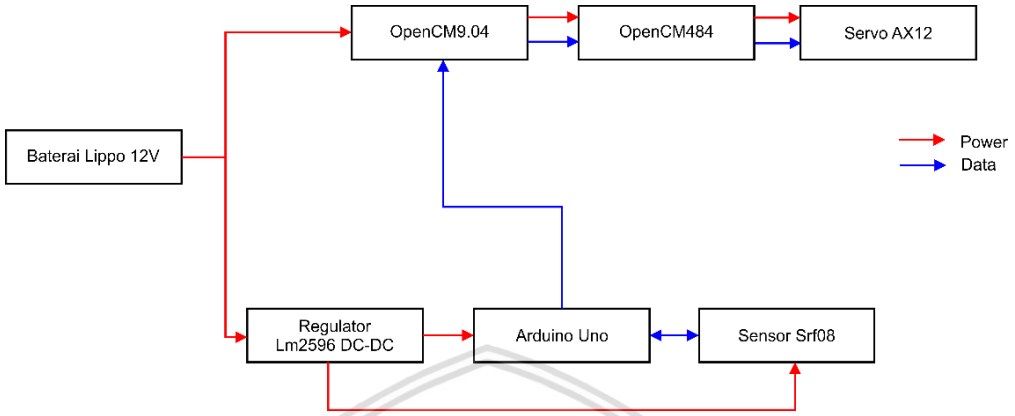
Berdasarkan metode penelitian diatas, perancangan sistem dilakukan setelah melakukan analisa kebutuhan sistem. Perancangan penelitian ini dilakukan untuk merencanakan tahapan-tahapan yang harus dilakukan dalam proses melakukan penelitian. Tahapan penelitian yang harus dilakukan antara lain perancangan perangkat lunak dan perangkat keras. Diharapkan dengan dilakukannya tahapan perancangan penelitian dapat memperlancar proses pelaksanaan penelitian. Gambaran secara umum perancangan sistem pada Gambar 5.1.



Gambar 5.1 Blog Diagram Perancangan Sistem

5.1.1 Perancangan Perangkat Keras

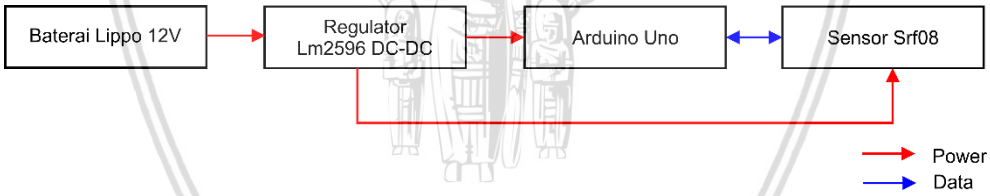
Perancangan pada perangkat keras secara keseluruhan dapat dilihat pada diagram blog pada Gambar 5.2.



Gambar 5.2 Skematik Keseluruhan Sistem

Pada skematik Gambar 5.2, garis merah merupakan jalur sumber tegangan dari sistem. Terdapat dua jalur tegangan yang dibutuhkan yaitu tegangan 12v yang dibutuhkan oleh OpemCM9.04, ekspansi OpenCM485 dan Servo AX-12 yang didapat langsung dari sumber tegangan baterai lippo 12v, serta tegangan 5Vvyang dibutuhkan oleh Arduino Uno dan sensor Srf08 yang didapat dari sumber tegangan baterai lipo 12v yang diturunkan menjadi 5v menggunakan Regulator LM2596.

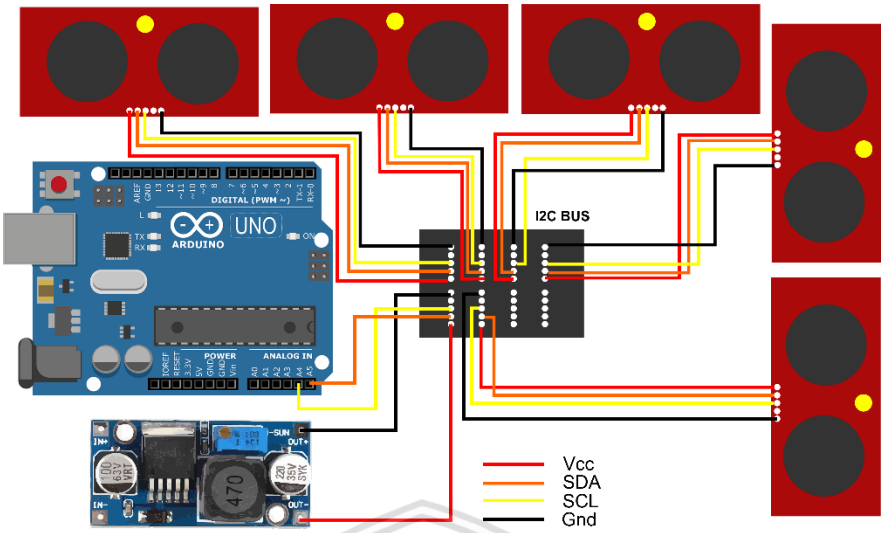
5.1.1.1 Perancangan Subsistem Sensor Srf08



Gambar 5.3 Subsistem Sensor Srf08

Penjelasan dari Gambar 5.3 yang merupakan gambar blok diagram subsistem sensor srf08:

1. Sumber tegangan dari baterai 12v diturunkan menjadi 5v menggunakan regulator LM2596. Kemudian diteruskan sebagai sumber tegangan Arduino Uno dan sensor Srf08 yang membutuhkan tegangan sebesar 5v. Skematik pembacaan sensor Srf08 dapat dilihat pada Gambar 5.4.
2. Arduino Uno membaca data dari sensor Srf08 yang kemudian diolah menggunakan logika *fuzzy*.



Gambar 5.4 Skematik Konfigurasi Pin Sensor Srf08 dengan Arduino

Sensor Srf08 menggunakan komunikasi serial I2C dalam pembacaan data sensor menggunakan jalur pin serial data (SDA) dan serial clock (SCL). Semua sensor dapat digunakan secara bersama dengan cara membedakan alamat pada sensor tersebut. Alamat *default* pada sensor tersebut adalah 0xE0 yang dapat dirubah kedalam 16 alamat yang tersedia yaitu E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC atau FE. Dengan membedakan alamat antar sensor maka data sensor dapat terbaca secara bergantian sesuai perintah dari kode program. Adapun alamat sensor dapat dikenali dengan cara menghitung led yang menyala pada saat awal sensor dinyalakan seperti pada Tabel 5.1.

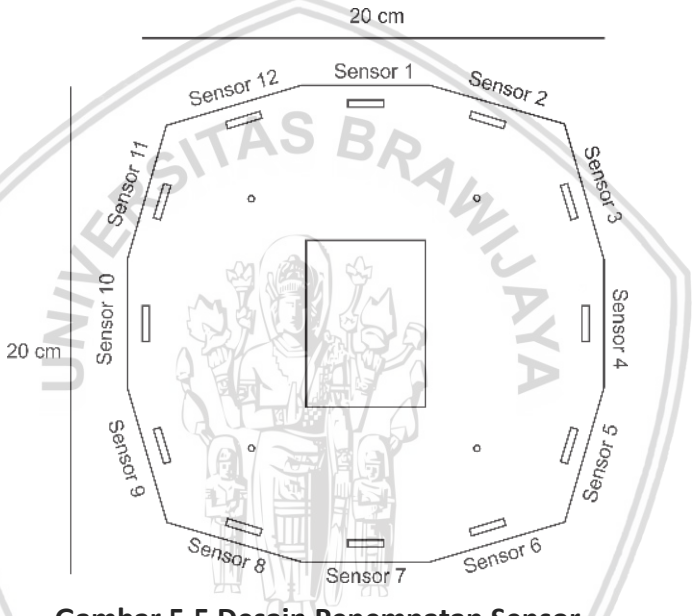
Tabel 5.1 Alamat Sensor Srf08

Address		Long Flash	Short Flashes
Decimal	Hex		
224	E0	1	0
226	E2	1	1
228	E4	1	2
230	E6	1	3
232	E8	1	4
234	EA	1	5
236	EC	1	6
238	EE	1	7
240	F0	1	8
242	F2	1	9
244	F4	1	10

246	F6	1	11
248	F8	1	12
250	FA	1	13
252	FC	1	14
254	FE	1	15

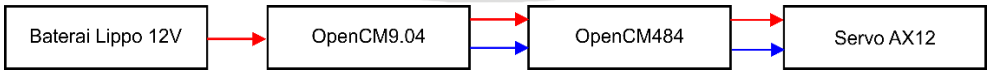
Sumber: (Robot-Elektronik, 2017)

Jumlah sensor yang digunakan pada sistem ini sebanyak 12 sensor yang dipasang melingkari robot. Sensor yang digunakan pada saat robot berjalan akan berganti sesuai dengan arah gerak robot. Adapun penempatan sensor dapat dilihat pada Gambar 5.5



Gambar 5.5 Desain Penempatan Sensor

5.1.1.2 Perancangan Subsistem OpenCM9.04

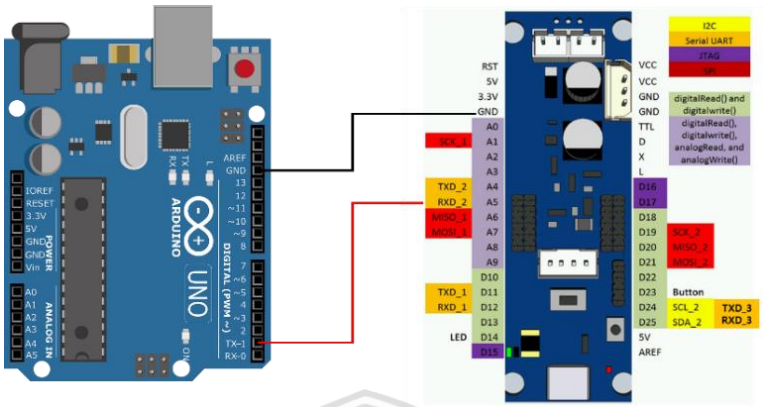


Gambar 5.6 Subsistem OpenCM9.04

Penjelasan dari gambar 5.6 yang merupakan gambar blok diagram subsistem OpenCM9.04:

1. OpenCM9.04 membutuhkan tegangan sebesar 12v yang didapat dari baterai.
2. OpenCM9.04 akan mengirimkan data ke ekspansi OpenCM485 dan diteruskan ke Servo AX-12

5.1.1.3 Perancangan Komunikasi Arduino Uno dengan OpenCM9.04



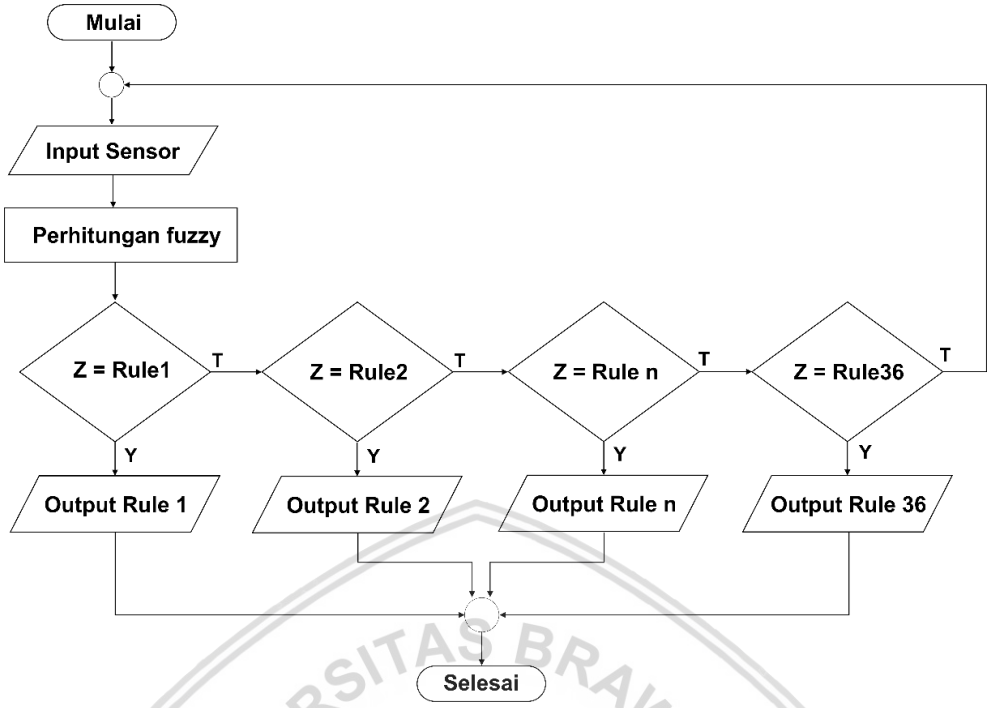
Gambar 5.7 Skematik Komunikasi Arduino Uno dengan OpenCM9.04

Penjelasan dari gambar 5.7 yang merupakan gambar skematik komunikasi Arduino Uno dengan OpenCM9.04:

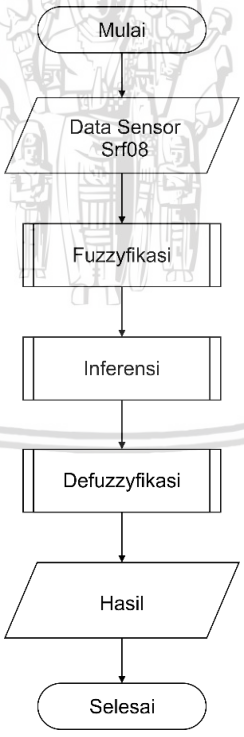
1. Arduino Uno mengirimkan data ke OpenCM9.04 menggunakan komunikasi serial UART dengan pin Tx Arduino terhubung dengan pin Rx OpenCM9.04.
2. OpenCM9.04 mempunyai tiga pin Rx dimana ketiganya dapat digunakan dengan menginisialisasi pin yang digunakan pada kode program.

5.1.2 Perancangan Perangkat Lunak

Perancangan pada perangkat lunak meliputi pembacaan sensor jarak Srf08 oleh mikrokontroller Arduino Uno dan kemudian data sensor tersebut diproses menggunakan metode *fuzzy* sugeno. Adapun perancangan perangkat lunak digambarkan pada Gambar 5.8. Tahapan-tahapan *fuzzy* sugeno dimulai dari membuat variable input, menentukan fungsi keanggotaan input dan membuat *rule fuzzy*. Hasil dari perhitungan *fuzzy* selanjutnya untuk menentukan output yang akan dikirim ke OpenCM9.04 untuk menggerakkan kaki robot. Adapun flowchart *fuzzy* digambarkan pada Gambar 5.9.



Gambar 5.8 Flowchart Perancangan Perangkat Lunak

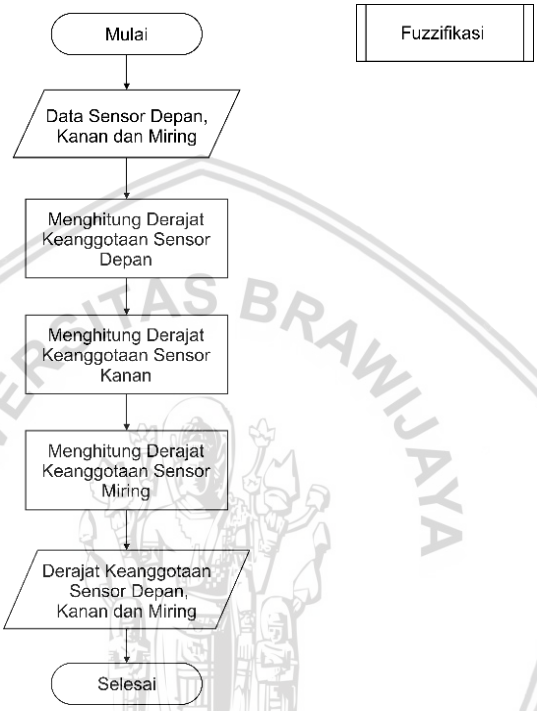


Gambar 5.9 *Flowchart Fuzzy*

Terdapat dua perancangan pada perangkat lunak yaitu perancangan pada manuver robot *omni direction* dan prancangan pada manuver robot memutar. Kedua perancangan tersebut memiliki proses yang sama, hanya berbeda pada *rules fuzzy* dan output yang dihasilkan.

5.1.2.1 Fuzzifikasi

Dalam perancangan logika *fuzzy*, langkah pertama yang dilakukan adalah proses fuzzifikasi yaitu proses pengubahan nilai tegas (Crisp) dalam fungsi keanggotaan *fuzzy* untuk menentukan derajat keangotaan variabel input *fuzzy* yaitu derajat keanggotaan sensor depan, sensor kanan dan sensor miring. Pada proses fuzzifikasi antara manuver robot *omni direction* dan manuver robot memutar mempunyai perancangan yang sama. Proses fuzzifikasi digambarkan pada diagram flow pada Gambar 5.10.



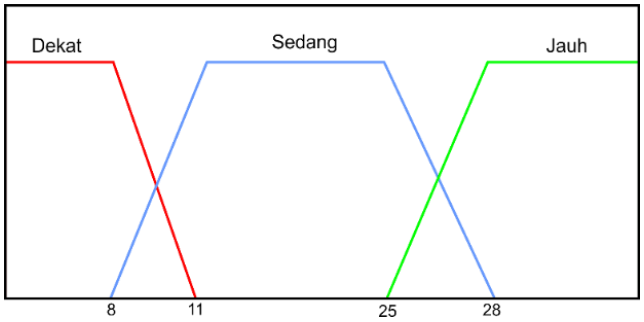
Gambar 5.10 Diagram Flow Fuzzifikasi

Gambar 5.10 merupakan gambar diagram alir dari fuzzifikasi dimulai dari menerima input sensor, yang terdiri dari sensor depan, sensor kanan dan sensor miring. Kemudian data tersebut digunakan untuk menghitung derajat keanggotaan jarak sensor depan, derajat keanggotaan jarak sensor kanan dan jarak keanggotaan sensor miring. Kemudian hasilnya akan diperoleh derajat keanggotaan sensor depan, kanan dan miring.

Berikut ini merupakan perhitungan derajat keanggotaan sensor depan, sensor kanan dan sensor miring.

a. Sensor Depan

Fungsi keanggotaan sensor depan ditunjukkan pada gambar 5.11



Gambar 5.11 Fungsi Keanggotaan Sensor Depan

Gambar 5.11 merupakan fungsi keanggotaan sensor depan. Terdapat tiga derajat keanggotaan yaitu dekat yaitu kurang dari 11cm, sedang yaitu antara 8cm hingga 28cm dan jauh yaitu diatas 28cm. Dari gambar 5.11 dapat dimasukkan ke dalam untuk mencari derajat keanggotaan representase kurva trapezium sehingga didapatkan perhitungan sebagai berikut:

Dekat(x) : $0 \quad : x > 11$

$\frac{11-x}{11-8} \quad : 8 < x \leq 11$

$1 \quad : x < 8$

Sedang(x) : $0 \quad : x \leq 8 \; atau \; x \geq 28$

$\frac{x-8}{11-8} \quad : 8 < x \leq 11$

$\frac{28-x}{28-25} \quad : 25 < x < 28$

$1 \quad : 11 < x \leq 25$

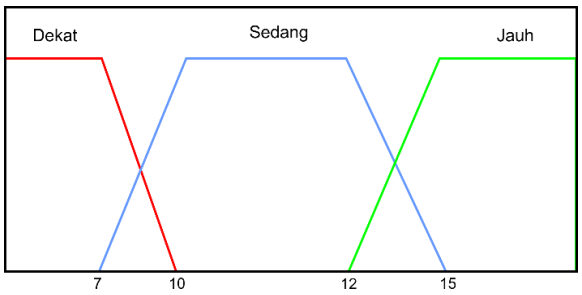
Jauh(x) : $0 \quad : x < 25$

$\frac{x-25}{28-25} \quad : 25 < x \leq 28$

$1 \quad : x > 28$

b. Sensor Kanan

Fungsi keanggotaan sensor kanan ditunjukkan pada Gambar 5.12.



Gambar 5.12 Fungsi Keanggotaan Sensor Kanan

Gambar 5.12 merupakan fungsi keanggotaan sensor kanan. Terdapat tiga derajat keanggotaan meliputi dekat yaitu kurang dari 10cm, sedang yaitu antara

7cm hingga 15cm dan jauh yaitu diatas 15cm. Dari Gambar 5.12 dapat dimasukkan ke dalam untuk mencari derajat keanggotaan representase kurva trapezium sehingga didapatkan perhitungan sebagai berikut:

Dekat(x) : $0 \quad : x > 10$

$\frac{10-x}{10-7} \quad : 7 < x \leq 10$

$1 \quad : x < 7$

Sedang(x) : $0 \quad : x \leq 7 \; atau \; x \geq 19$

$\frac{x-7}{10-7} \quad : 7 < x \leq 10$

$\frac{15-x}{15-12} \quad : 12 < x < 15$

$1 \quad : 10 < x \leq 12$

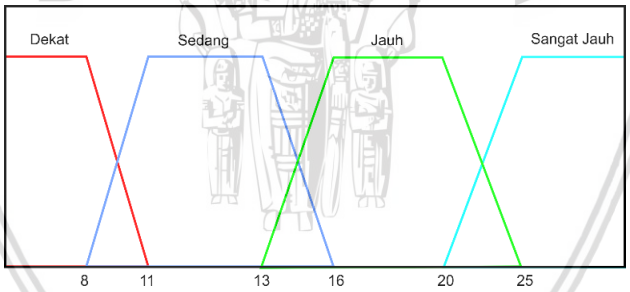
Jauh(x) : $0 \quad : x < 12$

$\frac{x-12}{15-12} \quad : 12 < x \leq 15$

$1 \quad : x > 15$

c. Sensor Miring

Fungsi keanggotaan sensor miring ditunjukkan pada gambar 5.13



Gambar 5.13 Fungsi Keanggotaan Sensor Miring

Gambar 5.13 merupakan fungsi keanggotaan sensor miring. Terdapat empat derajat keanggotaan meliputi dekat yaitu kurang dari 11cm, sedang yaitu antara 8cm hingga 16cm, jauh yaitu antara 13cm hingga 25cm dan sangat jauh yaitu diatas 20cm. Dari Gambar 5.13 dapat dimasukkan ke dalam untuk mencari derajat keanggotaan representase kurva trapezium sehingga didapatkan perhitungan sebagai berikut:

Dekat(x) : $0 \quad : x > 11$

$\frac{11-x}{11-8} \quad : 8 < x \leq 11$

$1 \quad : x < 8$

Sedang(x) : $0 \quad : x \leq 8$

$$\frac{x-8}{11-8} : 8 < x \leq 11$$

$$\frac{16-x}{16-13} : 13 < x < 16$$

$$1 : 11 < x \leq 13$$

Jauh(x) : $0 : x < 13$

$$\frac{x-13}{16-13} : 13 < x \leq 16$$

$$\frac{25-x}{25-20} : 20 < x \leq 25$$

$$1 : 16 < x \leq 20$$

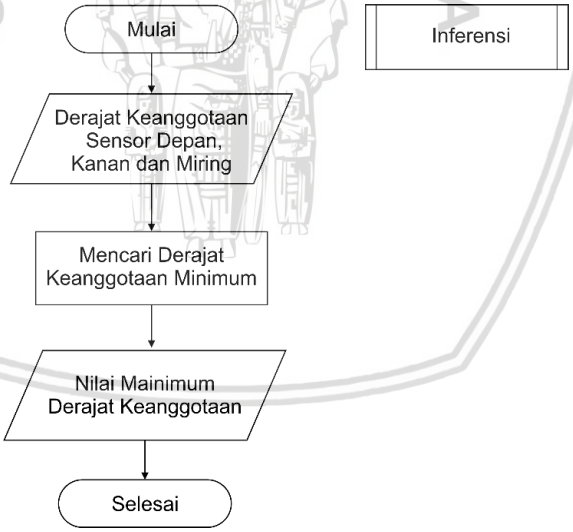
Sangat Jauh(x): $0 : x < 20$

$$\frac{x-20}{25-20} : 20 < x \leq 25$$

$$1 : x > 25$$

5.1.2.2 Inferensi

Setelah proses fuzzifikasi dilakukan, maka proses selanjutnya yakni inferensi. Proses inferensi digambarkan dengan diagram alir berikut pada Gambar 5.14.



Gambar 5.14 Diagram Alir Infereni

Diagram alir inferensi diatas digambarkan mulai dari data hasil fuzzifikasi berupa derajat keanggotaan sensor depan, kanan dan miring. Dari data keanggotaan tersebut dibandingkan antara semua derajat keanggotaan dan dicari nilai minimumnya. Adapun terdapat dua *rule* yang dibuat yaitu *rule* manuver robot *omni direction* dan *rule* manuver robot memutar. Tabel rule *fuzzy* pada manuver robot *omni direction* ditunjukkan pada Tabel 5.2 sedangkan tabel *rule* manuver robot memutar ditunjukkan pada Tabel 5.3.

Tabel 5.1 *Rule* Fuzzy Manuver Robot *Omni Direction*

Rules	Sensor Depan	Sensor Kanan	Sensor Miring	Output			
				Nilai	Kiri	Kanan	Delay
Rule 0	Dekat	Dekat	Dekat	1	Manuver Kiri		
Rule 1	Dekat	Dekat	Sedang	1	Manuver Kiri		
Rule 2	Dekat	Dekat	Jauh	1	Manuver Kiri		
Rule 3	Dekat	Dekat	Sangat Jauh	1	Manuver Kiri		
Rule 4	Dekat	Sedang	Dekat	1	Manuver Kiri		
Rule 5	Dekat	Sedang	Sedang	1	Manuver Kiri		
Rule 6	Dekat	Sedang	Jauh	1	Manuver Kiri		
Rule 7	Dekat	Sedang	Sangat Jauh	1	Manuver Kiri		
Rule 8	Dekat	Jauh	Dekat	1	Manuver Kiri		
Rule 9	Dekat	Jauh	Sedang	1	Manuver Kiri		
Rule 10	Dekat	Jauh	Jauh	1	Manuver Kiri		
Rule 11	Dekat	Jauh	Sangat Jauh	1	Manuver Kiri		
Rule 12	Sedang	Dekat	Dekat	1	Pelan	Cepat	120
Rule 13	Sedang	Dekat	Sedang	1	Pelan	Cepat	120
Rule 14	Sedang	Dekat	Jauh	1	Cepat	Cepat	120
Rule 15	Sedang	Dekat	Sangat Jauh	1	Cepat	Cepat	120
Rule 16	Sedang	Sedang	Dekat	1	Pelan	Cepat	120
Rule 17	Sedang	Sedang	Sedang	1	Cepat	Cepat	120
Rule 18	Sedang	Sedang	Jauh	1	Pelan	Cepat	120
Rule 19	Sedang	Sedang	Sangat Jauh	1	Pelan	Pelan	120
Rule 20	Sedang	Jauh	Dekat	1	Cepat	Pelan	120
Rule 21	Sedang	Jauh	Sedang	1	Cepat	Pelan	120
Rule 22	Sedang	Jauh	Jauh	1	Cepat	Pelan	120
Rule 23	Sedang	Jauh	Sangat Jauh	1	Manuver Kanan		
Rule 24	Jauh	Dekat	Dekat	1	Pelan	Cepat	80
Rule 25	Jauh	Dekat	Sedang	1	Pelan	Cepat	80
Rule 26	Jauh	Dekat	Jauh	1	Pelan	Cepat	80
Rule 27	Jauh	Dekat	Sangat Jauh	1	Pelan	Cepat	80
Rule 28	Jauh	Sedang	Dekat	1	Pelan	Cepat	80
Rule 29	Jauh	Sedang	Sedang	1	Cepat	Cepat	80
Rule 30	Jauh	Sedang	Jauh	1	Cepat	Pelan	80
Rule 31	Jauh	Sedang	Sangat Jauh	1	Pelan	Pelan	80
Rule 32	Jauh	Jauh	Dekat	1	Cepat	Pelan	80
Rule 33	Jauh	Jauh	Sedang	1	Cepat	Cepat	80
Rule 34	Jauh	Jauh	Jauh	1	Geser Kanan		
Rule 35	Jauh	Jauh	Sangat Jauh	1	Manuver Kanan		

Tabel 5.2 *Rule* Fuzzy Manuber Robot Memutar

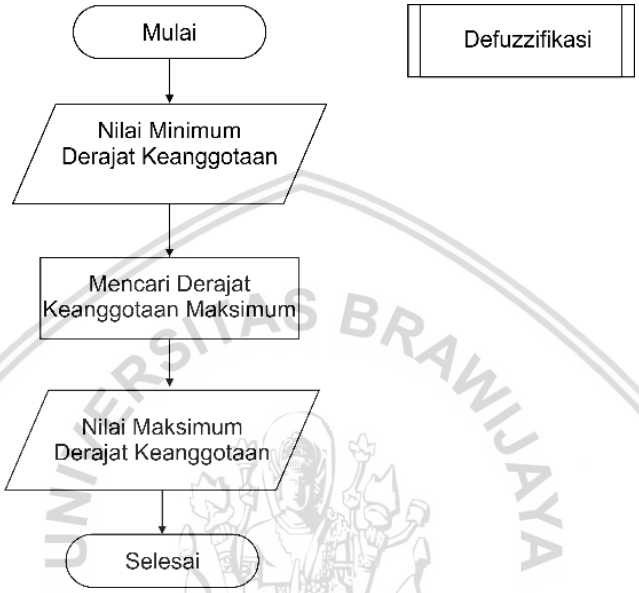
Rules	Sensor Depan	Sensor Kanan	Sensor Miring Depan	Output			
				Nilai	*Speed Left*	*Speed Right*	*Delay*
Rule 0	Dekat	Dekat	Dekat	1	Belok Kiri		
Rule 1	Dekat	Dekat	Sedang	1	Belok Kiri		
Rule 2	Dekat	Dekat	Jauh	1	Belok Kiri		
Rule 3	Dekat	Dekat	Sangat Jauh	1	Belok Kiri		
Rule 4	Dekat	Sedang	Dekat	1	Belok Kiri		
Rule 5	Dekat	Sedang	Sedang	1	Belok Kiri		
Rule 6	Dekat	Sedang	Jauh	1	Belok Kiri		
Rule 7	Dekat	Sedang	Sangat Jauh	1	Belok Kiri		
Rule 8	Dekat	Jauh	Dekat	1	Belok Kiri		
Rule 9	Dekat	Jauh	Sedang	1	Belok Kiri		
Rule 10	Dekat	Jauh	Jauh	1	Belok Kiri		
Rule 11	Dekat	Jauh	Sangat Jauh	1	Belok Kiri		
Rule 12	Sedang	Dekat	Dekat	1	Pelan	Cepat	120
Rule 13	Sedang	Dekat	Sedang	1	Pelan	Cepat	120
Rule 14	Sedang	Dekat	Jauh	1	Pelan	Cepat	120
Rule 15	Sedang	Dekat	Sangat Jauh	1	Pelan	Pelan	120
Rule 16	Sedang	Sedang	Dekat	1	Pelan	Cepat	120
Rule 17	Sedang	Sedang	Sedang	1	Cepat	Cepat	120
Rule 18	Sedang	Sedang	Jauh	1	Pelan	Cepat	120
Rule 19	Sedang	Sedang	Sangat Jauh	1	Pelan	Pelan	120
Rule 20	Sedang	Jauh	Dekat	1	Cepat	Pelan	120
Rule 21	Sedang	Jauh	Sedang	1	Cepat	Pelan	120
Rule 22	Sedang	Jauh	Jauh	1	Cepat	Pelan	120
Rule 23	Sedang	Jauh	Sangat Jauh	1	Cepat	Pelan	120
Rule 24	Jauh	Dekat	Dekat	1	Pelan	Cepat	80
Rule 25	Jauh	Dekat	Sedang	1	Pelan	Cepat	80
Rule 26	Jauh	Dekat	Jauh	1	Pelan	Cepat	80
Rule 27	Jauh	Dekat	Sangat Jauh	1	Pelan	Cepat	80
Rule 28	Jauh	Sedang	Dekat	1	Pelan	Cepat	80
Rule 29	Jauh	Sedang	Sedang	1	Cepat	Cepat	80
Rule 30	Jauh	Sedang	Jauh	1	Cepat	Pelan	80
Rule 31	Jauh	Sedang	Sangat Jauh	1	Cepat	Pelan	80
Rule 32	Jauh	Jauh	Dekat	1	Cepat	Pelan	80
Rule 33	Jauh	Jauh	Sedang	1	Cepat	Cepat	80
Rule 34	Jauh	Jauh	Jauh	1	Geser Kanan		
Rule 35	Jauh	Jauh	Sangat Jauh	1	Cepat	Pelan	80

Dalam *rule fuzzy* terdapat 3 output yaitu *speed left*, *speed right* dan *delay*. *Speed left* dan *speed right* merupakan kecepatan pada gerakan kaki kanan maupun kiri yang mepunyai 2 jenis kecepatan yaitu pelan dan cepat. Dengan perbedaan

kecepatan gerakan kaki maka akan membuat jalan robot sedikit berbelok yang berguna untuk mengatur jarak robot dengan dinding. *Delay* pada *rule fuzzy* berfungsi untuk melakukan jeda pada setiap *step motion* sehingga kecepatan robot dapat diatur dengan *delay* tersebut.

5.1.2.3 Defuzzifikasi

Setelah proses inferensi selesai, maka proses selanjutnya adalah defuzifikasi yang digambarkan dengan diagram alir pada Gambar 5.15.

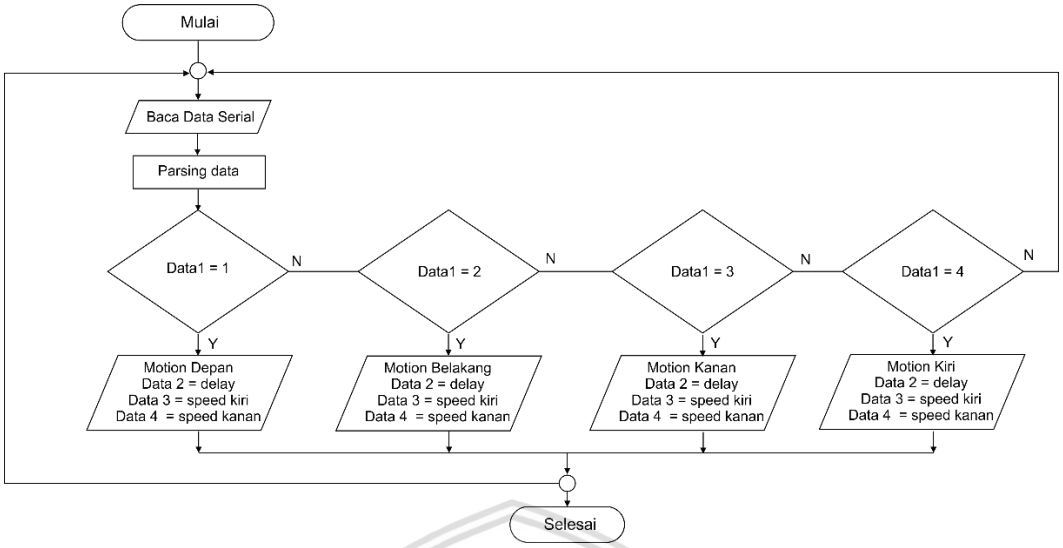


Gambar 5.15 Diagram Alir Defuzzifikasi

Proses defuzzifikasi merupakan salah satu proses pemetaan himpunan *fuzzy* ke dalam himpunan tegas (crisp) dengan cara mencari nilai maksimum. Setelah mendapatkan nilai α dari metode MIN-MAX pada setiap variabel akan dievaluasi pada setiap *rule* atau aturan untuk mencari nilai terbesar.

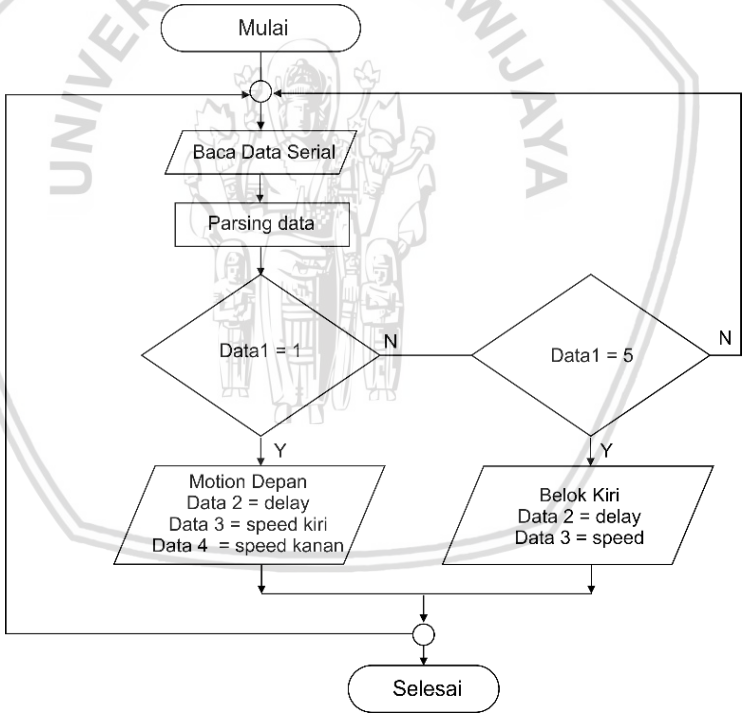
5.1.2.4 Perancangan Pergerakan Robot

Pergerakan robot dimulai pada saat OpenCM9.04 menerima data dari Arduino Uno kemudian akan dilakukan parsing data sehingga didapatkan empat variabel dimana variabel pertama digunakan untuk memanggi fungsi pergerakan robot. Terdapat dua perancangan pada pergerakan robot yaitu perancangan pada manuver robot *omni direction* dan perancangan pada manuver robot memutar. Pada manuver robot *omni direction* terdapat empat jenis pergerakan yaitu pergerakan maju, pergerakan mundur, pergerakan kanan dan pergerakan kiri yang digambarkan dengan diagram alir pada Gambar 5.16.



Gambar 5.16 Diagram Alir Pergerakan Menuver Robot *Omni Direction*

Sedangkan perancangan manuver robot memutar digambarkan dengan diagram alir seperti pada Gambar 5.17.



Gambar 5.17 Diagram Alir Pergerakan Manuver Robot Memutar

5.2 Implementasi Sistem

Setelah melalui tahap perancangan sistem, maka selanjutnya adalah implementasi sistem sesuai hasil perancangan. Implementasi sistem yang dilakukan adalah implementasi perangkat keras dan implementasi perangkat lunak.

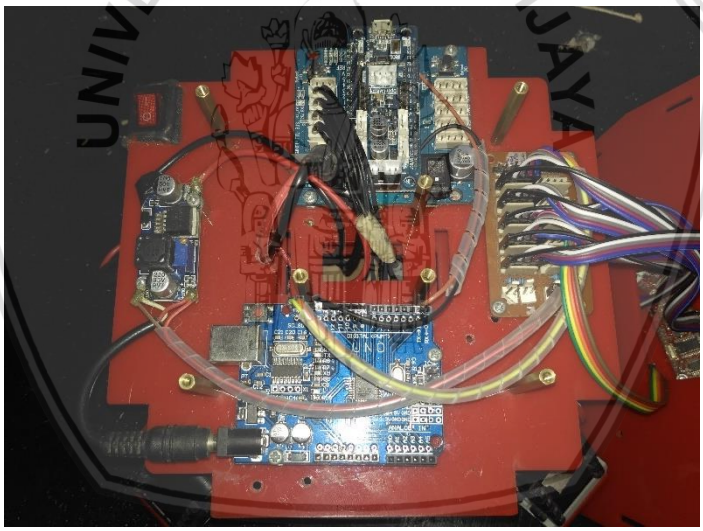
5.2.1 Batasan Implementasi

Batasan-batasan dalam penelitian ini adalah sebagai berikut :

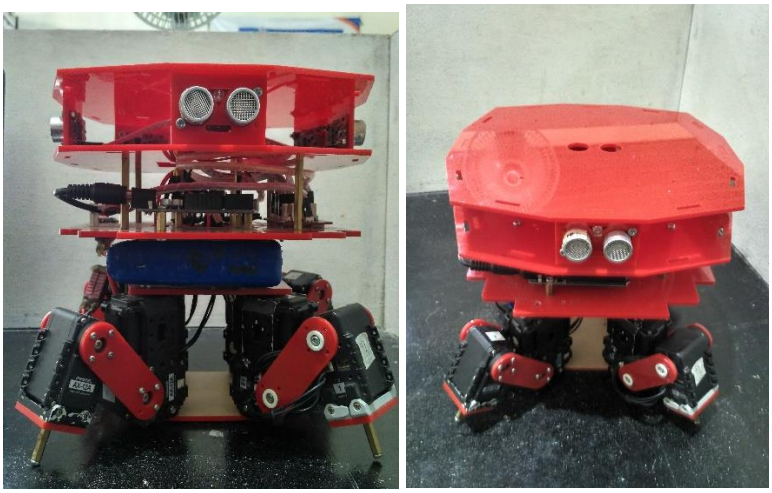
1. Robot dengan panjang × lebar × tinggi tidak lebih dari 30cm × 30cm × 27cm.
2. Bahan yang digunakan pada tubuh robot adalah akrilik dengan ketebalan 3mm.
3. Pengolahan algoritma wall following dan metode fuzzy diolah dan dikomputasi di Arduino Uno.
4. OpenCM9.04 dan ekspansi OpenCM485 hanya berfungsi untuk menggerakkan servo dan pergerakan robot yang telah dibuat.

5.2.2 Implementasi Perangkat Keras

Pada implementasi perangkat keras dilakukan pemasangan komponen-komponen perangkat keras pada body robot quadruped yang dapat dilihat pada Gambar 5.18 dan Gambar 5.19.



Gambar 5.18 Penempatan Komponen Pada Robot

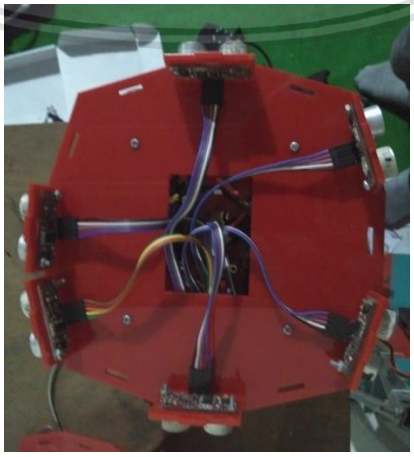


Gambar 5.19 Robot Tampak Depan

5.2.2.1 Implementasi Subsistem Sensor Srf08

Pada implementasi subsistem sensor Srf08 dilakukan implementasi sesuai dengan perancangan subsistem sensor Srf08. Hasil implementasi dapat dilihat pada Gambar 5.20 dan Gambar 5.21.

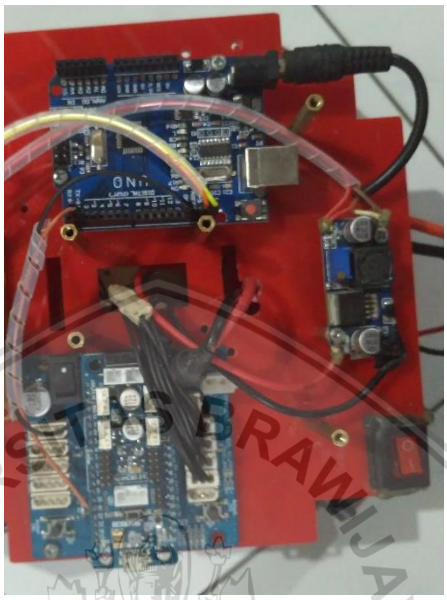
Gambar 5.20 Implementasi Subsistem Sensor Srf08



Gambar 5.21 Implementasi Desain Penempatan Sensor Srf08

5.2.2.2 Implementasi Subsistem Komunikasi Arduino Uno dengan OpenCM9.04

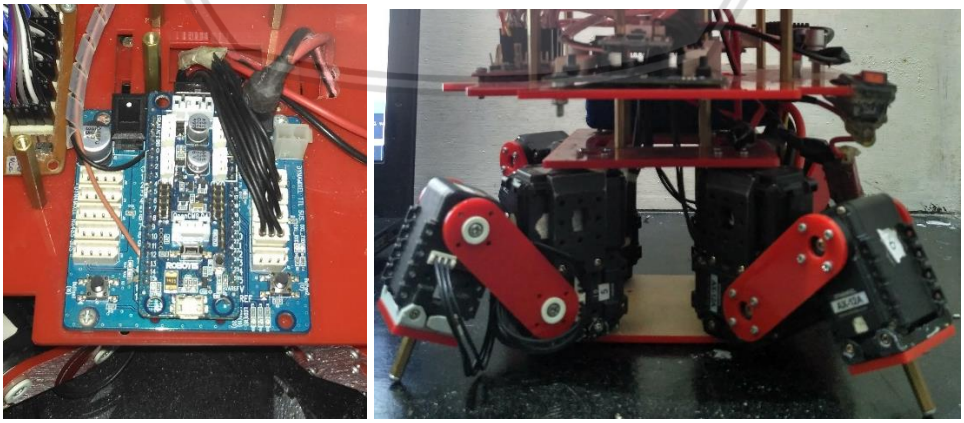
Pada implementasi subsistem komunikasi Arduino Uno dengan OpenCM9.04 dilakukan implementasi sesuai dengan perancangan subsistem komunikasi Arduino Uno dengan OpenCM9.04. Hasil implementasi dapat dilihat pada Gambar 5.22.



Gambar 5.22 Implementasi Subsistem Komunikasi Arduino Uno dengan OpenCM9.04

5.2.2.3 Implementasi Subsistem OpenCM9.04

Pada implementasi subsistem OpenCM9.04 dilakukan implementasi sesuai dengan perancangan subsistem OpenCM9.04. Hasil implementasi dapat dilihat pada gambar 5.23.



Gambar 5.23 Implementasi Subsistem Komunikasi Arduino Uno dengan OpenCM9.04

5.2.3 Implementasi Perangkat Lunak

Pada implementasi perangkat lunak dilakukan beberapa implementasi beberapa bagian yaitu perhitungan *fuzzy*, komunikasi dan motion. Dalam pemrograman arduino dibutuhkan deklarasi variabel dan linclude library yang dibutuhkan pada program. Kode program deklarasi variabel dan program main adalah seperti pada Tabel 5.4.

Tabel 5.3 Program Main Sistem

Nomor	Kode Program
1	#include <Array.h>
2	#include <Wire.h>
3	#define A 112
4	#define B 113
5	#define C 114
6	#define D 117
7	#define E 118
8	#define F 117
9	#define G 117
10	#define H 118
11	
12	char direct = 'N'; // inisialisasi direction robot
13	char left = 'W';
14	int function = 1;
15	int functionLeft = 4;
16	int functionRight = 3;
17	//--variable keanggotaan input
18	float sd[3]; //sensor depan
19	float sk[3]; //sensor kanan
20	float smd[4]; //sensor miring depan
21	//float smb[3]; // sensor miring belakang
22	//-------------------
23	int reading = 0;
24	//--deklarasi variable input sensor
25	int sensorDepan = 0;
26	int sensorKanan = 0;
27	int sensorMiringDepan = 0;
28	//---------------------------
29	float rules[36];
30	float defu_final;
31	float defu[12];
32	
33	
34	int srf(int address /*,int srfnumber*/) {
35	Wire.beginTransmission(address);
36	Wire.write(0x00);
37	Wire.write(0x54);//command measure in "cm" (0x54)
38	Wire.endTransmission();
39	delay(70);
40	Wire.beginTransmission(address);
41	Wire.write(0x02);
42	Wire.endTransmission();
43	Wire.requestFrom(address, 2);
44	if (2 <= Wire.available()) {
45	reading = Wire.read();
46	reading = reading << 8;

```
47      reading |= Wire.read();  }
48    return reading;
49  }
50
51  void setup() {
52    Wire.begin();
53    Serial.begin(250000);
54    delay(3000);
55  }
56
57  void loop() {
58    Serial.println(direct);
59    if (direct == 'N') {
60      sensorDepan = srf(A);
61      sensorMiringDepan = srf(B);
62      sensorKanan = srf(C);
63    } else if (direct == 'E') {
64      sensorDepan = srf(C);
65      sensorMiringDepan = srf(D);
66      sensorKanan = srf(E);
67    } else if (direct == 'S') {
68      sensorDepan = srf(E);
69      sensorMiringDepan = srf(F);
70      sensorKanan = srf(G);
71    } else if (direct == 'W') {
72      sensorDepan = srf(G);
73      sensorMiringDepan = srf(H);
74      sensorKanan = srf(A);
75    }
76    //-----fuzzyfikasi------
77  keanggotaan_sensor_depan_uji(sensorDepan);
78  keanggotaan_sensor_kanan(sensorKanan);
79  keanggotaan_sensor_miring_depan(sensorMiringDepan);
80    //--------inferensi-----------------
81    infer(); //rule fuzzy
82    defuzzyfikasi(); //fungsi defuzzyfikasi
83    output1();
84    //output2();
85  }
86
```

Terdapat dua program output sesuai dengan perancangan sistem yang telah dilakukan yaitu output program manuver robot *omni direction* yang dapat dilihat pada Tabel 5.5 dan output program manuve robot memutar yang dapat dilihat pada Tabel 5.6.

Tabel 5.4 Kode Program Output Manuver Robot *Omni Direction*

Nomor	Kode Program
1	`void output1() {`
2	`  if (defu_final == rules[0]) {`
3	`    Serial.println("Rules 0");`
4	`    manuverKiri();`
5	`  } else if (defu_final == rules[1]) {`
6	`    Serial.println("Rules 1");`
7	`    manuverKiri();`

```
  } else if (defu_final == rules[2]) {
    Serial.println("Rules 2");
    manuverKiri();
  } else if (defu_final == rules[3]) {
    Serial.println("Rules 3");
    manuverKiri();
  } else if (defu_final == rules[4]) {
    Serial.println("Rules 4");
    manuverKiri();
  } else if (defu_final == rules[5]) {
    Serial.println("Rules 5");
    manuverKiri();
  } else if (defu_final == rules[6]) {
    Serial.println("Rules 6");
    manuverKiri();
  } else if (defu_final == rules[7]) {
    Serial.println("Rules 7");
    manuverKiri();
  } else if (defu_final == rules[8]) {
    Serial.println("Rules 8");
    manuverKiri();
  } else if (defu_final == rules[9]) {
    Serial.println("Rules 9");
    manuverKiri();
  } else if (defu_final == rules[10]) {
    Serial.println("Rules 10");
    manuverKiri();
  } else if (defu_final == rules[11]) {
    Serial.println("Rules 11");
    manuverKiri();
  } else if (defu_final == rules[12]) {
    Serial.println("Rules 12");
    Serial.print("*");
    Serial.print(function);
    Serial.println(",120,140,1023#");
  } else if (defu_final == rules[13]) {
    Serial.println("Rules 13");
    Serial.print("*");
    Serial.print(function);
    Serial.println(",120,140,1023#");
  } else if (defu_final == rules[14]) {
    Serial.println("Rules 14");
    Serial.print("*");
    Serial.print(function);
    Serial.println(",120,1023,1023#");
  } else if (defu_final == rules[15]) {
    Serial.println("Rules 15");
    Serial.print("*");
    Serial.print(function);
    Serial.println(",120,1023,1023#");
  } else if (defu_final == rules[16]) {
    Serial.println("Rules 16");
    Serial.print("*");
    Serial.print(function);
    Serial.println(",120,140,1023#");
  } else if (defu_final == rules[17]) {
    Serial.println("Rules 17");
    Serial.print("*");
    Serial.print(function);
```

```
66      Serial.println(",120,1023,1023#");
67    } else if (defu_final == rules[18]) {
68      Serial.println("Rules 18");
69      Serial.print("*");
70      Serial.print(function);
71      Serial.println(",120,1023,140#");
72    } else if (defu_final == rules[19]) {
73      Serial.println("Rules 19");
74      Serial.print("*");
75      Serial.print(function);
76      Serial.println(",120,1023,140#");
77    } else if (defu_final == rules[20]) {
78      Serial.println("Rules 20");
79      Serial.print("*");
80      Serial.print(function);
81      Serial.println(",120,1023,130#");
82    } else if (defu_final == rules[21]) {
83      Serial.println("Rules 21");
84      Serial.print("*");
85      Serial.print(function);
86      Serial.println(",120,1023,130#");
87    } else if (defu_final == rules[22]) {
88      Serial.println("Rules 22");
89      Serial.print("*");
90      Serial.print(function);
91      Serial.println(",120,1023,130#");
92    } else if (defu_final == rules[23]) {
93      Serial.println("Rules 23");
94      for (int i = 0; i < 10; i++) {
95        Serial.print("*");
96        Serial.print(function);
97        Serial.println(",120,150,1023#");
98        delay(300);
99      }
100     manuverKanan();
101     while (sensorKanan > 30) {
102       Serial.print("*");
103       Serial.print(function);
104       Serial.println(",120,150,1023#");
105       delay(300);
106     }
107   } else if (defu_final == rules[24]) {
108     Serial.println("Rules 24");
109     Serial.print("*");
110     Serial.print(function);
111     Serial.println(",80,170,1023#");
112   } else if (defu_final == rules[25]) {
113     Serial.println("Rules 25");
114     Serial.print("*");
115     Serial.print(function);
116     Serial.println(",80,160,1023#");
117   } else if (defu_final == rules[26]) {
118     Serial.println("Rules 26");
119     Serial.print("*");
120     Serial.print(function);
121     Serial.println(",80,160,1023#");
122   } else if (defu_final == rules[27]) {
123     Serial.println("Rules 27");
124     Serial.print("*");
```

```
125       Serial.print(function);
126       Serial.println(",120,1023,1023#");
127     } else if (defu_final == rules[28]) {
128       Serial.println("Rules 28");
129       Serial.print("*");
130       Serial.print(function);
131       Serial.println(",80,170,1023#");
132     } else if (defu_final == rules[29]) {
133       Serial.println("Rules 29");
134       Serial.print("*");
135       Serial.print(function);
136       Serial.println(",80,1023,170#");
137     } else if (defu_final == rules[30]) {
138       Serial.println("Rules 30");
139       Serial.print("*");
140       Serial.print(function);
141       Serial.println(",80,1023,150#");
142     } else if (defu_final == rules[31]) {
143       Serial.println("Rules 31");
144       Serial.print("*");
145       Serial.print(function);
146       Serial.println(",80,1023,150#");
147     } else if (defu_final == rules[32]) {
148       Serial.println("Rules 32");
149       Serial.print("*");
150       Serial.print(function);
151       Serial.println(",80,1023,170#");
152     } else if (defu_final == rules[33]) {
153       Serial.println("Rules 33");
154       Serial.print("*");
155       Serial.print(function);
156       Serial.println(",80,1023,170#");
157     } else if (defu_final == rules[34]) {
158       Serial.println("Rules 34");
159       geserKanan();
160     } else if (defu_final == rules[35]) {
161       Serial.println("Rules 35");
162       for (int i = 0; i < 10; i++) {
163         Serial.print("*");
164         Serial.print(function);
165         Serial.println(",120,140,1023#");
166         delay(300);
167       }
168       manuverKanan();
169       for (int i = 0; i < 10; i++) {
170         Serial.print("*");
171         Serial.print(function);
172         Serial.println(",120,140,1023#");
173         delay(300);
174       }
175     } else {
176       Serial.println("null");
177     }
178     delay(80);
179   }
180
181   void manuverKanan() {
182     if (direct == 'N') {
183       direct = 'E';
```

```
    function = 3;
  } else if (direct == 'E') {
    direct = 'S';
    function = 2;
  } else if (direct == 'S') {
    direct = 'W';
    function = 4;
  } else if (direct == 'W') {
    direct = 'N';
    function = 1;
  }
}

void manuverKiri() {
  if (direct == 'N') {
    direct = 'W';
    function = 4;
  } else if (direct == 'W') {
    direct = 'S';
    function = 2;
  } else if (direct == 'S') {
    direct = 'E';
    function = 3;
  } else if (direct == 'E') {
    direct = 'N';
    function = 1;
  }
}

void geserKanan() {
  if (direct == 'N') {
    functionRight = 3;
    Serial.print("*");
    Serial.print(functionRight);
    Serial.println(",120,1023,1023#");
  } else if (direct == 'W') {
    functionRight = 1;
    Serial.print("*");
    Serial.print(functionRight);
    Serial.println(",120,1023,1023#");
  } else if (direct == 'S') {
    functionRight = 4;
    Serial.print("*");
    Serial.print(functionRight);
    Serial.println(",120,1023,1023#");
  } else if (direct == 'E') {
    functionRight = 2;
    Serial.print("*");
    Serial.print(functionRight);
    Serial.println(",120,1023,1023#");
  }
}
```

Tabel 5.5 Kode Program Output Manuver Robot *Memutar*

```
void output_manuver_memutar() {
  if (defu_final == rules[0]) {
    Serial.println("Rules 0");
```

```
4       for (int i = 0; i < 10; i++) {
5         Serial.println("*5,80,300#");
6         delay(200);
7       }
8     } else if (defu_final == rules[1]) {
9       Serial.println("Rules 1");
10      for (int i = 0; i < 10; i++) {
11        Serial.println("*5,80,300#");
12        delay(200);
13      }
14    } else if (defu_final == rules[2]) {
15      Serial.println("Rules 2");
16      for (int i = 0; i < 10; i++) {
17        Serial.println("*5,80,300#");
18        delay(200);
19      }
20    } else if (defu_final == rules[3]) {
21      Serial.println("Rules 3");
22      for (int i = 0; i < 10; i++) {
23        Serial.println("*5,80,300#");
24        delay(200);
25      }
26    } else if (defu_final == rules[4]) {
27      Serial.println("Rules 4");
28      for (int i = 0; i < 10; i++) {
29        Serial.println("*5,80,300#");
30        delay(200);
31      }
31    } else if (defu_final == rules[5]) {
32      Serial.println("Rules 5");
33      for (int i = 0; i < 10; i++) {
34        Serial.println("*5,80,300#");
35        delay(200);
36      }
37    } else if (defu_final == rules[6]) {
38      Serial.println("Rules 6");
39      for (int i = 0; i < 10; i++) {
40        Serial.println("*5,80,300#");
41        delay(200);
42      }
43    } else if (defu_final == rules[7]) {
44      Serial.println("Rules 7");
45      for (int i = 0; i < 10; i++) {
46        Serial.println("*5,80,300#");
47        delay(200);
48      }
49    } else if (defu_final == rules[8]) {
50      Serial.println("Rules 8");
51      for (int i = 0; i < 10; i++) {
52        Serial.println("*5,80,300#");
53        delay(200);
54      }
55    } else if (defu_final == rules[9]) {
56      Serial.println("Rules 9");
57      for (int i = 0; i < 10; i++) {
58        Serial.println("*5,80,300#");
59        delay(200);
60      }
61    } else if (defu_final == rules[10]) {
```

```
62      Serial.println("Rules 10");
63      for (int i = 0; i < 10; i++) {
64        Serial.println("*5,80,300#");
65        delay(200);
66      }
67    } else if (defu_final == rules[11]) {
68      Serial.println("Rules 11");
69      for (int i = 0; i < 10; i++) {
70        Serial.println("*5,80,300#");
71        delay(200);
72      }
73    } else if (defu_final == rules[12]) {
74      Serial.println("Rules 12");
75      Serial.print("*");
76      Serial.print(function);
77      Serial.println(",120,120,1000#");
78    } else if (defu_final == rules[13]) {
79      Serial.println("Rules 13");
80      Serial.print("*");
81      Serial.print(function);
82      Serial.println(",120,130,1000#");
83    } else if (defu_final == rules[14]) {
84      Serial.println("Rules 14");
85      Serial.print("*");
86      Serial.print(function);
87      Serial.println(",120,130,1000#");
88    } else if (defu_final == rules[15]) {
89      Serial.println("Rules 15");
90      Serial.print("*");
91      Serial.print(function);
92      Serial.println(",120,130,1000#");
93    } else if (defu_final == rules[16]) {
94      Serial.println("Rules 16");
95      Serial.print("*");
96      Serial.print(function);
97      Serial.println(",120,130,1000#");
98    } else if (defu_final == rules[17]) {
99      Serial.println("Rules 17");
100     Serial.print("*");
101     Serial.print(function);
102     Serial.println(",120,130,1000#");
103   } else if (defu_final == rules[18]) {
104     Serial.println("Rules 18");
105     Serial.print("*");
106     Serial.print(function);
107     Serial.println(",120,130,1000#");
108   } else if (defu_final == rules[19]) {
109     Serial.println("Rules 19");
110     Serial.print("*");
111     Serial.print(function);
112     Serial.println(",120,130,1000#");
113   } else if (defu_final == rules[20]) {
114     Serial.println("Rules 20");
115     Serial.print("*");
116     Serial.print(function);
117     Serial.println(",120,130,1000#");
118   } else if (defu_final == rules[21]) {
119     Serial.println("Rules 21");
120     Serial.print("*");
```

```
121       Serial.print(function);
122       Serial.println(",120,130,1000#");
123     } else if (defu_final == rules[22]) {
124       Serial.println("Rules 22");
125       Serial.print("*");
126       Serial.print(function);
127       Serial.println(",120,130,1000#");
128     } else if (defu_final == rules[23]) {
129       Serial.println("Rules 23");
130       Serial.print("*");
131       Serial.print(function);
132       Serial.println(",120,1000,100#");
133     } else if (defu_final == rules[24]) {
134       Serial.println("Rules 24");
135       Serial.print("*");
136       Serial.print(function);
137       Serial.println(",70,200,1000#");
138     } else if (defu_final == rules[25]) {
139       Serial.println("Rules 25");
140       Serial.print("*");
141       Serial.print(function);
142       Serial.println(",80,200,1000#");
143     } else if (defu_final == rules[26]) {
144       Serial.println("Rules 26");
145       Serial.print("*");
146       Serial.print(function);
147       Serial.println(",80,200,1000#");
148     } else if (defu_final == rules[27]) {
149       Serial.println("Rules 27");
150       Serial.print("*");
151       Serial.print(function);
152       Serial.println(",120,200,1000#");
153     } else if (defu_final == rules[28]) {
154       Serial.println("Rules 28");
155       Serial.print("*");
156       Serial.print(function);
157       Serial.println(",80,1000,1000#");
158     } else if (defu_final == rules[29]) {
159       Serial.println("Rules 29");
160       Serial.print("*");
161       Serial.print(function);
162       Serial.println(",80,1000,1000#");
163     } else if (defu_final == rules[30]) {
164       Serial.println("Rules 30");
165       Serial.print("*");
166       Serial.print(function);
167       Serial.println(",80,1000,200#");
168     } else if (defu_final == rules[31]) {
169       Serial.println("Rules 31");
170       Serial.print("*");
171       Serial.print(function);
172       Serial.println(",120,1000,200#");
173     } else if (defu_final == rules[32]) {
174       Serial.println("Rules 32");
175       Serial.print("*");
176       Serial.print(function);
177       Serial.println(",80,1000,200#");
178     } else if (defu_final == rules[33]) {
179       Serial.println("Rules 33");
```

```
180       Serial.print("*");
181       Serial.print(function);
182       Serial.println(",80,1000,200#");
183     } else if (defu_final == rules[34]) {
184       Serial.println("Rules 34");
185       geserKanan();
186     } else if (defu_final == rules[35]) {
187       Serial.println("Rules 35");
188       Serial.print("*");
189       Serial.print(function);
190       Serial.println(",120,1023,60#");
191     } else {
192       Serial.println("null");
193     }
194     delay(80);
195   }
```

5.2.3.1 Fuzzifikasi

Implementasi fuzzifikasi dilakukan berdasarkan hasil perancangan fuzzifikasi. Kode program untuk fuzzifikasi berupa keanggotaan sensor depan, sensor kanan dan sensor miring adalah pada Tabel 5.7.

Tabel 5.6 Program Fuzzifikasi

Nomor	Kode Program
1	`float keanggotaan_sensor_depan(float x) {`
2	`  Serial.print("x depan = ");`
3	`  Serial.println(x);`
4	`  //dekat`
5	`  if (x <= 8) {`
6	`    sd[0] = 1;`
7	`  } else if (x > 8 && x <= 12) {`
8	`    sd[0] = ((12 - x) / (12 - 8));`
9	`  } else if (x > 12) {`
10	`    sd[0] = 0;`
11	`  }`
12	`  //sedang`
13	`  if (x <= 8) {`
14	`    sd[1] = 0;`
15	`  } else if (x > 8 && x <= 12) {`
16	`    sd[1] = ((x - 8) / (12 - 8));`
17	`  } else if (x > 12 && x <= 25) {`
18	`    sd[1] = 1;`
19	`  } else if (x > 25 && x <= 29) {`
20	`    sd[1] = ((29 - x) / (29 - 25));`
21	`  } else if (x >= 29) {`
22	`    sd[1] = 0;`
23	`  }`
24	`  //jauh`
25	`  if (x <= 25) {`
26	`    sd[2] = 0;`
27	`  } else if (x > 25 && x <= 29) {`
28	`    sd[2] = ((x - 25) / (29 - 25));`
29	`  } else if (x >= 29) {`
30	`    sd[2] = 1;`

```
  }
}

float keanggotaan_sensor_kanan(float x) {
  Serial.print("x kanan = ");
  Serial.println(x);
  //dekat
  if (x <= 7) {
    sk[0] = 1;
  } else if (x > 7 && x <= 10) {
    sk[0] = ((10 - x) / (10 - 7));
  } else if (x > 10) {
    sk[0] = 0;
  }
  //sedang
  if (x <= 7) {
    sk[1] = 0;
  } else if (x > 7 && x <= 10) {
    sk[1] = ((x - 7) / (10 - 7));
  } else if (x > 10 && x <= 12) {
    sk[1] = 1;
  } else if (x > 12 && x <= 15) {
    sk[1] = ((15 - x) / (15 - 12));
  } else if (x >= 15) {
    sk[1] = 0;
  }
  //jauh
  if (x <= 12) {
    sk[2] = 0;
  } else if (x > 12 && x <= 15) {
    sk[2] = ((x - 12) / (15 - 12));
  } else if (x >= 15) {
    sk[2] = 1;
  }

}

float keanggotaan_sensor_miring_depan(float x) {
  Serial.print("x miring = ");
  Serial.println(x);
  //dekat
  if (x <= 8) {
    smd[0] = 1;
  } else if (x > 8 && x <= 11) {
    smd[0] = ((11 - x) / (11 - 8));
  } else if (x > 11) {
    smd[0] = 0;
  }
  //sedang
  if (x <= 8) {
    smd[1] = 0;
  } else if (x > 8 && x <= 11) {
    smd[1] = ((x - 8) / (11 - 8));
  } else if (x > 11 && x <= 13) {
    smd[1] = 1;
  } else if (x > 13 && x <= 16) {
    smd[1] = ((16 - x) / (16 - 13));
  } else if (x > 16) {
    smd[1] = 0;
```

```
90    }
91    //jauh
92    if (x <= 13) {
93      smd[2] = 0;
94    } else if (x > 13 && x <= 16) {
95      smd[2] = ((x - 13) / (16 - 13));
96    } else if ((x > 16) && (x <= 20)) {
97      smd[2] = 1;
98    } else if (x < 20 && x <= 25) {
99      smd[2] = ((25 - x) / (25 - 20));
100   } else if (x >= 25) {
101     smd[2] = 0;
102     }
103   //sangat jauh
104   if (x <= 20) {
105     smd[3] = 0;
106   } else if (x < 20 && x <= 25) {
107     smd[3] = ((x - 20) / (25 - 20));
108   } else if (x >= 25) {
109     smd[3] = 1;
110   }
111 }
```

5.2.3.2 Inferensi

Implementasi inferensi dilakukan berdasarkan hasil perancangan inferensi. Kode program untuk inferensi adalah pada Tabel 5.8.

Tabel 5.7 Kode Program Inferensi

Nomor	Kode Program
1	`void infer() {`
2	`  int h = 0;`
3	`  for (int i = 0; i < 3; i++) {`
4	`    sd[i];`
5	`    for (int j = 0; j < 3; j++) {`
6	`      sk[j];`
7	`      for (int k = 0; k < 4; k++) {`
8	`        smd[k];`
9	`        rules[h] = min(sd[i], min(sk[j], smd[k]));`
10	`        h += 1;`
11	`      }}}}`

5.2.3.3 Defuzzifikasi

Implementasi defuzzifikasi dilakukan berdasarkan hasil perancangan defuzzifikasi yang merupakan output dari *fuzzy*. Kode program defuzzifikasi dapat dilihat pada Table 5.9.

Tabel 5.8 Kode Program Defuzzifikasi

Nomor	Kode Program
1	`void defuzzyfikasi() {`
2	`  const byte size = 36;`
3	`  float rawArray[size] = {rules[0], rules[1], rules[2],`
4	`rules[3], rules[4], rules[5],ules[6], rules[7], rules[8],`

```
rules[9],  rules[10],  rules[11],  rules[12],  rules[13],
rules[14],  rules[15],  rules[16],  rules[17],  rules[18],
rules[19],  rules[20],  rules[21],  rules[22],  rules[23],
rules[24],  rules[25],  rules[26],  rules[27],  rules[28],
rules[29],  rules[30],  rules[31],  rules[32],  rules[33],
rules[34], rules[35]
};

  Array<float> array = Array<float>(rawArray, size);
  defu_final = array.getMax();
}
```

5.2.3.4 Pergerakan Robot

Untuk menggerakkan kaki robot dibutuhkan beberapa langkah agar robot dapat bergerak dengan baik menggunakan 4 kaki. Sebelum OpenCM9.04 melakukan gerakan servo terdapat program untuk menerima data dan melakukan parsing data untuk menentukan pergerakan yang dapat dilihat pada table 5.10. Selain itu juga terdapat kode program untuk gerakan manuver robot *omni direction* yaitu pergerakan maju yang dapat dilihat pada Tabel 5.11, pergerakan mundur pada Tabel 5.12, pergerakan kanan pada Tabel 5.13 dan pergerakan kiri pada Tabel 5.14. Sedangkan pada manuver robot memutar terdapat dua pergerakan yaitu pergerakan maju yang dapat dilihat pada Tabel 5.11 dan pergerakan belok kiri yang dapat dilihat pada Tabel 5.15.

Tabel 5.9 Pergerakan Robot

Nomor	Kode Program
1	`#define DXL_BUS_Serial3 3`
2	`#define ID_NUM1 1`
3	`#define ID_NUM2 2`
4	`#define ID_NUM3 3`
5	`#define ID_NUM4 4`
6	`#define ID_NUM5 5`
7	`#define ID_NUM6 6`
8	`#define ID_NUM7 7`
9	`#define ID_NUM8 8`
10	`#define ID_NUM9 9`
11	`#define ID_NUM10 10`
12	`#define ID_NUM11 11`
13	`#define ID_NUM12 12`
14	
15	`Dynamixel Dxl(DXL_BUS_Serial3);`
16	
17	`String dataIn;`
18	`String dt[10];`
19	`int i;`
20	`boolean parsing = false;`
21	
22	`char command;`
23	`String string;`
24	`char data;`
25	`char x;`

```
26
27  void setup() {
28    Dxl.begin(3);
29    Dxl.jointMode(ID_NUM1);
30    Dxl.jointMode(ID_NUM2);
31    Dxl.jointMode(ID_NUM3);
32    Dxl.jointMode(ID_NUM4);
33    Dxl.jointMode(ID_NUM5);
34    Dxl.jointMode(ID_NUM6);
35    Dxl.jointMode(ID_NUM7);
36    Dxl.jointMode(ID_NUM8);
37    Dxl.jointMode(ID_NUM9);
38    Dxl.jointMode(ID_NUM10);
39    Dxl.jointMode(ID_NUM11);
40    Dxl.jointMode(ID_NUM12);
41    Stand();
42    SerialUSB.begin(250000);
43    Serial1.begin(250000);
44    delay(2000);
45  }
46
47  void loop(){
48    if (Serial1.available() > 0) {
49      char inChar = (char)Serial1.read();
50      dataIn += inChar;
51      //    SerialUSB.println(dataIn);
52      if (inChar == '\n') {
53        parsing = true;
54      }
55    }
56    if (parsing) {
57      //SerialUSB.print("masuk ke ");
58      //SerialUSB.println(x++);
59      parsingData();
60      parsing = false;
61      dataIn = "";
62      int function = dt[0].toInt();
63      int dely = dt[1].toInt();
64      int speedL = dt[2].toInt();
65      int speedR = dt[3].toInt();
66      if (function == 1) {
67        forward(dely, speedL, speedR);
68      } else if (function == 2) {
69        backward(dely, speedL, speedR);
70      } else if (function == 3) {
71        right(dely, speedL, speedR);
72      } else if (function == 4) {
73        left(dely, speedL, speedR);
74      }else if (function == 5) {
75        turn_left(dely, speedL);
76      }
77      else{
78        Serial.println("nothing");
79      }
80      SerialUSB.print("\n");
81    }
82  }
83  void parsingData() {
84    int j = 0;
```

```
85    //kirim data yang telah diterima sebelumnya
86    SerialUSB.print("data masuk : ");
87    SerialUSB.print(dataIn);
88  //  SerialUSB.print("\n");
89
90    dt[j] = "";
91    for (i = 1; i < dataIn.length(); i++) {
92      if ((dataIn[i] == '#') || (dataIn[i] == ',')) {
93        j++;
94        dt[j] = "";
95      } else {
96        dt[j] = dt[j] + dataIn[i];
97      }
98    }
99  }
100 void Stand(){
101 Dxl.goalPosition(ID_NUM1,512);
102 Dxl.goalPosition(ID_NUM2,562);
103 Dxl.goalPosition(ID_NUM3,562);
104 Dxl.goalPosition(ID_NUM4,512);
105 Dxl.goalPosition(ID_NUM5,562);
106 Dxl.goalPosition(ID_NUM6,562);
107 Dxl.goalPosition(ID_NUM7,512);
108 Dxl.goalPosition(ID_NUM8,462);
109 Dxl.goalPosition(ID_NUM9,462);
110 Dxl.goalPosition(ID_NUM10,512);
111 Dxl.goalPosition(ID_NUM11,462);
112 Dxl.goalPosition(ID_NUM12,462);
113 }
```

Table 5.10 Kode Program Pergerakan Maju

Nomor	Kode Program
1	`void forward(int delai,int spdl, int spdr) {`
2	`  step_f1(spdl, spdr);`
3	`  delay(delai);`
4	`  step_f2(spdl, spdr);`
5	`  delay(delai);`
6	`  step_f3(spdl, spdr);`
7	`  delay(delai);`
8	`  step_f4(spdl, spdr);`
9	`  delay(delai);`
10	`}`
11	`void step_f1(int spdl, int spdr) {`
12	`  Dxl.setPosition(ID_NUM1, 462, spdr);`
13	`  Dxl.setPosition(ID_NUM2, 632, spdr);`
14	`  Dxl.setPosition(ID_NUM3, 612, spdr);`
15	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
16	`  Dxl.setPosition(ID_NUM5, 612, spdl);`
17	`  Dxl.setPosition(ID_NUM6, 612, spdl);`
18	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
19	`  Dxl.setPosition(ID_NUM8, 412, spdr);`
20	`  Dxl.setPosition(ID_NUM9, 412, spdr);`
21	`  Dxl.setPosition(ID_NUM10, 562, spdl);`
22	`  Dxl.setPosition(ID_NUM11, 382, spdl);`
23	`  Dxl.setPosition(ID_NUM12, 412, spdl);`
24	`}`

```
25  void step_f2(int spdl, int spdr) {
26    Dxl.setPosition(ID_NUM1, 462, spdr);
27    Dxl.setPosition(ID_NUM2, 612, spdr);
28    Dxl.setPosition(ID_NUM3, 612, spdr);
29    Dxl.setPosition(ID_NUM4, 512, spdl);
30    Dxl.setPosition(ID_NUM5, 612, spdl);
31    Dxl.setPosition(ID_NUM6, 612, spdl);
32    Dxl.setPosition(ID_NUM7, 512, spdr);
33    Dxl.setPosition(ID_NUM8, 412, spdr);
34    Dxl.setPosition(ID_NUM9, 412, spdr);
35    Dxl.setPosition(ID_NUM10, 562, spdl);
36    Dxl.setPosition(ID_NUM11, 412, spdl);
37    Dxl.setPosition(ID_NUM12, 412, spdl);
38  }
39
40  void step_f3(int spdl, int spdr) {
41    Dxl.setPosition(ID_NUM1, 512, spdr);
42    Dxl.setPosition(ID_NUM2, 612, spdr);
43    Dxl.setPosition(ID_NUM3, 612, spdr);
44    Dxl.setPosition(ID_NUM4, 562, spdl);
45    Dxl.setPosition(ID_NUM5, 632, spdl);
46    Dxl.setPosition(ID_NUM6, 612, spdl);
47    Dxl.setPosition(ID_NUM7, 462, spdr);
48    Dxl.setPosition(ID_NUM8, 382, spdr);
49    Dxl.setPosition(ID_NUM9, 412, spdr);
50    Dxl.setPosition(ID_NUM10, 512, spdl);
51    Dxl.setPosition(ID_NUM11, 412, spdl);
52    Dxl.setPosition(ID_NUM12, 412, spdl);
53  }
54  void step_f4(int spdl, int spdr) {
55    Dxl.setPosition(ID_NUM1, 512, spdr);
56    Dxl.setPosition(ID_NUM2, 612, spdr);
57    Dxl.setPosition(ID_NUM3, 612, spdr);
58    Dxl.setPosition(ID_NUM4, 562, spdl);
59    Dxl.setPosition(ID_NUM5, 612, spdl);
60    Dxl.setPosition(ID_NUM6, 612, spdl);
61    Dxl.setPosition(ID_NUM7, 462, spdr);
62    Dxl.setPosition(ID_NUM8, 412, spdr);
63    Dxl.setPosition(ID_NUM9, 412, spdr);
64    Dxl.setPosition(ID_NUM10, 512, spdl);
65    Dxl.setPosition(ID_NUM11, 412, spdl);
66    Dxl.setPosition(ID_NUM12, 412, spdl);
67  }
```

Tabel 5.11 Kode Program Pergerakan Mundur

Nomor	Kode Program
1	`void backward(int delai,int spdl, int spdr) {`
2	`  step_b1(spdl, spdr);`
3	`  delay(delai);`
4	`  step_b2(spdl, spdr);`
5	`  delay(delai);`
6	`  step_b3(spdl, spdr);`
7	`  delay(delai);`
8	`  step_b4(spdl, spdr);`
9	`  delay(delai);`
10	`}`

```
11  void step_b1(int spdl, int spdr) {
12    Dxl.setPosition(ID_NUM1, 562, spdl);
13    Dxl.setPosition(ID_NUM2, 632, spdl);
14    Dxl.setPosition(ID_NUM3, 612, spdl);
15    Dxl.setPosition(ID_NUM4, 512, spdr);
16    Dxl.setPosition(ID_NUM5, 612, spdr);
17    Dxl.setPosition(ID_NUM6, 612, spdr);
18    Dxl.setPosition(ID_NUM7, 512, spdl);
19    Dxl.setPosition(ID_NUM8, 412, spdl);
20    Dxl.setPosition(ID_NUM9, 412, spdl);
21    Dxl.setPosition(ID_NUM10, 462, spdr);
22    Dxl.setPosition(ID_NUM11, 382, spdr);
23    Dxl.setPosition(ID_NUM12, 412, spdr);
24  }
25  void step_b2(int spdl, int spdr) {
26    Dxl.setPosition(ID_NUM1, 562, spdl);
27    Dxl.setPosition(ID_NUM2, 612, spdl);
28    Dxl.setPosition(ID_NUM3, 612, spdl);
29    Dxl.setPosition(ID_NUM4, 512, spdr);
30    Dxl.setPosition(ID_NUM5, 612, spdr);
31    Dxl.setPosition(ID_NUM6, 612, spdr);
32    Dxl.setPosition(ID_NUM7, 512, spdl);
33    Dxl.setPosition(ID_NUM8, 412, spdl);
34    Dxl.setPosition(ID_NUM9, 412, spdl);
35    Dxl.setPosition(ID_NUM10, 462, spdr);
36    Dxl.setPosition(ID_NUM11, 412, spdr);
37    Dxl.setPosition(ID_NUM12, 412, spdr);
38  }
39  void step_b3(int spdl, int spdr) {
40    Dxl.setPosition(ID_NUM1, 512, spdl);
41    Dxl.setPosition(ID_NUM2, 612, spdl);
42    Dxl.setPosition(ID_NUM3, 612, spdl);
43    Dxl.setPosition(ID_NUM4, 462, spdr);
44    Dxl.setPosition(ID_NUM5, 632, spdr);
45    Dxl.setPosition(ID_NUM6, 612, spdr);
46    Dxl.setPosition(ID_NUM7, 562, spdl);
47    Dxl.setPosition(ID_NUM8, 382, spdl);
48    Dxl.setPosition(ID_NUM9, 412, spdl);
49    Dxl.setPosition(ID_NUM10, 512, spdr);
50    Dxl.setPosition(ID_NUM11, 412, spdr);
51    Dxl.setPosition(ID_NUM12, 412, spdr);
52  }
53  void step_b4(int spdl, int spdr) {
54    Dxl.setPosition(ID_NUM1, 512, spdl);
55    Dxl.setPosition(ID_NUM2, 612, spdl);
56    Dxl.setPosition(ID_NUM3, 612, spdl);
57    Dxl.setPosition(ID_NUM4, 462, spdr);
58    Dxl.setPosition(ID_NUM5, 612, spdr);
59    Dxl.setPosition(ID_NUM6, 612, spdr);
60    Dxl.setPosition(ID_NUM7, 562, spdl);
61    Dxl.setPosition(ID_NUM8, 412, spdl);
62    Dxl.setPosition(ID_NUM9, 412, spdl);
63    Dxl.setPosition(ID_NUM10, 512, spdr);
64    Dxl.setPosition(ID_NUM11, 412, spdr);
65    Dxl.setPosition(ID_NUM12, 412, spdr);
66  }
```

Tabel 5.12 Kode Program Pergerakan Kanan

Nomor	Kode Program
1	`void right(int delai,int spdl, int spdr) {`
2	`  step_r1(spdl, spdr);`
3	`  delay(delai);`
4	`  step_r2(spdl, spdr);`
5	`  delay(delai);`
6	`  step_r3(spdl, spdr);`
7	`  delay(delai);`
8	`  step_r4(spdl, spdr);`
9	`  delay(delai);`
10	`}`
11	`void step_r1(int spdl, int spdr) {`
12	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
13	`  Dxl.setPosition(ID_NUM2, 612, spdl);`
14	`  Dxl.setPosition(ID_NUM3, 612, spdl);`
15	`  Dxl.setPosition(ID_NUM4, 562, spdl);`
16	`  Dxl.setPosition(ID_NUM5, 632, spdl);`
17	`  Dxl.setPosition(ID_NUM6, 612, spdl);`
18	`  Dxl.setPosition(ID_NUM7, 462, spdr);`
19	`  Dxl.setPosition(ID_NUM8, 382, spdr);`
20	`  Dxl.setPosition(ID_NUM9, 412, spdr);`
21	`  Dxl.setPosition(ID_NUM10, 512, spdr);`
22	`  Dxl.setPosition(ID_NUM11, 412, spdr);`
23	`  Dxl.setPosition(ID_NUM12, 412, spdr);`
24	`}`
25	`void step_r2(int spdl, int spdr) {`
26	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
27	`  Dxl.setPosition(ID_NUM2, 612, spdl);`
28	`  Dxl.setPosition(ID_NUM3, 612, spdl);`
29	`  Dxl.setPosition(ID_NUM4, 562, spdl);`
30	`  Dxl.setPosition(ID_NUM5, 612, spdl);`
31	`  Dxl.setPosition(ID_NUM6, 612, spdl);`
32	`  Dxl.setPosition(ID_NUM7, 462, spdr);`
33	`  Dxl.setPosition(ID_NUM8, 412, spdr);`
34	`  Dxl.setPosition(ID_NUM9, 412, spdr);`
35	`  Dxl.setPosition(ID_NUM10, 512, spdr);`
36	`  Dxl.setPosition(ID_NUM11, 412, spdr);`
37	`  Dxl.setPosition(ID_NUM12, 412, spdr);`
38	`}`
39	`void step_r3(int spdl, int spdr) {`
40	`  Dxl.setPosition(ID_NUM1, 562, spdl);`
41	`  Dxl.setPosition(ID_NUM2, 632, spdl);`
42	`  Dxl.setPosition(ID_NUM3, 612, spdl);`
43	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
44	`  Dxl.setPosition(ID_NUM5, 612, spdl);`
45	`  Dxl.setPosition(ID_NUM6, 612, spdl);`
46	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
47	`  Dxl.setPosition(ID_NUM8, 412, spdr);`
48	`  Dxl.setPosition(ID_NUM9, 412, spdr);`
49	`  Dxl.setPosition(ID_NUM10, 462, spdr);`
50	`  Dxl.setPosition(ID_NUM11, 382, spdr);`
51	`  Dxl.setPosition(ID_NUM12, 412, spdr);`
52	`}`
53	`void step_r4(int spdl, int spdr) {`
54	`  Dxl.setPosition(ID_NUM1, 562, spdl);`
55	`  Dxl.setPosition(ID_NUM2, 612, spdl);`

```
56    Dxl.setPosition(ID_NUM3, 612, spdl);
57    Dxl.setPosition(ID_NUM4, 512, spdl);
58    Dxl.setPosition(ID_NUM5, 612, spdl);
59    Dxl.setPosition(ID_NUM6, 612, spdl);
60    Dxl.setPosition(ID_NUM7, 512, spdr);
61    Dxl.setPosition(ID_NUM8, 412, spdr);
62    Dxl.setPosition(ID_NUM9, 412, spdr);
63    Dxl.setPosition(ID_NUM10, 462, spdr);
64    Dxl.setPosition(ID_NUM11, 412, spdr);
65    Dxl.setPosition(ID_NUM12, 412, spdr);
66  }
```

Tabel 5.13 Kode Program Pergerakan Kiri

Nomor	Kode Program

```
1   void left(int delai,int spdl, int spdr) {
2     step_l1(spdl, spdr);
3     delay(delai);
4     step_l2(spdl, spdr);
5     delay(delai);
6     step_l3(spdl, spdr);
7     delay(delai);
8     step_l4(spdl, spdr);
9     delay(delai);
10  }
11
12  void step_l1(int spdl, int spdr) {
13    Dxl.setPosition(ID_NUM1, 512, spdr);
14    Dxl.setPosition(ID_NUM2, 612, spdr);
15    Dxl.setPosition(ID_NUM3, 612, spdr);
16    Dxl.setPosition(ID_NUM4, 462, spdr);
17    Dxl.setPosition(ID_NUM5, 632, spdr);
18    Dxl.setPosition(ID_NUM6, 612, spdr);
19    Dxl.setPosition(ID_NUM7, 562, spdl);
20    Dxl.setPosition(ID_NUM8, 382, spdl);
21    Dxl.setPosition(ID_NUM9, 412, spdl);
22    Dxl.setPosition(ID_NUM10, 512, spdl);
23    Dxl.setPosition(ID_NUM11, 412, spdl);
24    Dxl.setPosition(ID_NUM12, 412, spdl);
25  }
26
27  void step_l2(int spdl, int spdr) {
28    Dxl.setPosition(ID_NUM1, 512, spdr);
29    Dxl.setPosition(ID_NUM2, 612, spdr);
30    Dxl.setPosition(ID_NUM3, 612, spdr);
31    Dxl.setPosition(ID_NUM4, 462, spdr);
32    Dxl.setPosition(ID_NUM5, 612, spdr);
33    Dxl.setPosition(ID_NUM6, 612, spdr);
34    Dxl.setPosition(ID_NUM7, 562, spdl);
35    Dxl.setPosition(ID_NUM8, 412, spdl);
36    Dxl.setPosition(ID_NUM9, 412, spdl);
37    Dxl.setPosition(ID_NUM10, 512, spdl);
38    Dxl.setPosition(ID_NUM11, 412, spdl);
39    Dxl.setPosition(ID_NUM12, 412, spdl);
40  }
41
42  void step_l3(int spdl, int spdr) {
43    Dxl.setPosition(ID_NUM1, 462, spdr);
44    Dxl.setPosition(ID_NUM2, 632, spdr);
```

```
45    Dxl.setPosition(ID_NUM3, 612, spdr);
46    Dxl.setPosition(ID_NUM4, 512, spdr);
47    Dxl.setPosition(ID_NUM5, 612, spdr);
48    Dxl.setPosition(ID_NUM6, 612, spdr);
49    Dxl.setPosition(ID_NUM7, 512, spdl);
50    Dxl.setPosition(ID_NUM8, 412, spdl);
51    Dxl.setPosition(ID_NUM9, 412, spdl);
52    Dxl.setPosition(ID_NUM10, 562, spdl);
53    Dxl.setPosition(ID_NUM11, 382, spdl);
54    Dxl.setPosition(ID_NUM12, 412, spdl);
55  }
56
57  void step_l4(int spdl, int spdr) {
58    Dxl.setPosition(ID_NUM1, 462, spdr);
59    Dxl.setPosition(ID_NUM2, 612, spdr);
60    Dxl.setPosition(ID_NUM3, 612, spdr);
61    Dxl.setPosition(ID_NUM4, 512, spdr);
62    Dxl.setPosition(ID_NUM5, 612, spdr);
63    Dxl.setPosition(ID_NUM6, 612, spdr);
64    Dxl.setPosition(ID_NUM7, 512, spdl);
65    Dxl.setPosition(ID_NUM8, 412, spdl);
66    Dxl.setPosition(ID_NUM9, 412, spdl);
67    Dxl.setPosition(ID_NUM10, 562, spdl);
68    Dxl.setPosition(ID_NUM11, 412, spdl);
69    Dxl.setPosition(ID_NUM12, 412, spdl);
70  }
```

Tabel 5.14 Kode Program Pergerakan Belok Kiri

Nomor	Kode Program
1	`void turn_right(int delai,int spd) {`
2	`  // for (int i=0; i <= 1; i++){`
3	`    step_tla(spd);`
4	`    delay(delai);`
5	`    step_tlb(spd);`
6	`    delay(delai);`
7	`    step_tlc(spd);`
8	`    delay(delai);`
9	`    step_tld(spd);`
10	`    delay(delai);`
11	`    step_tle(spd);`
12	`    delay(delai);`
13	`  }`
14	`  void step_tla(int spd) {`
15	`    Dxl.setPosition(ID_NUM1, 512, spd);`
16	`    Dxl.setPosition(ID_NUM2, 612, spd);`
17	`    Dxl.setPosition(ID_NUM3, 612, spd);`
18	`    Dxl.setPosition(ID_NUM4, 512, spd);`
19	`    Dxl.setPosition(ID_NUM5, 612, spd);`
20	`    Dxl.setPosition(ID_NUM6, 612, spd);`
21	`    Dxl.setPosition(ID_NUM7, 512, spd);`
22	`    Dxl.setPosition(ID_NUM8, 412, spd);`
23	`    Dxl.setPosition(ID_NUM9, 412, spd);`
24	`    Dxl.setPosition(ID_NUM10, 512, spd);`
25	`    Dxl.setPosition(ID_NUM11, 412, spd);`
26	`    Dxl.setPosition(ID_NUM12, 412, spd);`
27	`  }`

```
28  void step_tlb(int spd) {
29    Dxl.setPosition(ID_NUM1, 512, spd);
30    Dxl.setPosition(ID_NUM2, 662, spd);
31    Dxl.setPosition(ID_NUM3, 662, spd);
32    Dxl.setPosition(ID_NUM4, 512, spd);
33    Dxl.setPosition(ID_NUM5, 612, spd);
34    Dxl.setPosition(ID_NUM6, 612, spd);
35    Dxl.setPosition(ID_NUM7, 512, spd);
36    Dxl.setPosition(ID_NUM8, 412, spd);
37    Dxl.setPosition(ID_NUM9, 412, spd);
38    Dxl.setPosition(ID_NUM10, 512, spd);
39    Dxl.setPosition(ID_NUM11, 362, spd);
40    Dxl.setPosition(ID_NUM12, 362, spd);
41  }
42  void step_tlc(int spd) {
43    Dxl.setPosition(ID_NUM1, 562, spd);
44    Dxl.setPosition(ID_NUM2, 612, spd);
45    Dxl.setPosition(ID_NUM3, 612, spd);
46    Dxl.setPosition(ID_NUM4, 512, spd);
47    Dxl.setPosition(ID_NUM5, 612, spd);
48    Dxl.setPosition(ID_NUM6, 612, spd);
49    Dxl.setPosition(ID_NUM7, 512, spd);
50    Dxl.setPosition(ID_NUM8, 412, spd);
51    Dxl.setPosition(ID_NUM9, 412, spd);
52    Dxl.setPosition(ID_NUM10, 562, spd);
53    Dxl.setPosition(ID_NUM11, 412, spd);
54    Dxl.setPosition(ID_NUM12, 412, spd);
55  }
56  void step_tld(int spd) {
57    Dxl.setPosition(ID_NUM1, 462, spd);
58    Dxl.setPosition(ID_NUM2, 612, (spd));
59    Dxl.setPosition(ID_NUM3, 612, spd);
60    Dxl.setPosition(ID_NUM4, 512, spd);
61    Dxl.setPosition(ID_NUM5, 662, (spd));
62    Dxl.setPosition(ID_NUM6, 662, spd);
63    Dxl.setPosition(ID_NUM7, 512, spd);
64    Dxl.setPosition(ID_NUM8, 362, (spd));
65    Dxl.setPosition(ID_NUM9, 362, spd);
66    Dxl.setPosition(ID_NUM10, 462, spd);
67    Dxl.setPosition(ID_NUM11, 412, (spd));
68    Dxl.setPosition(ID_NUM12, 412, spd);
69  }
70
71  void step_tle(int spd) {
72    Dxl.setPosition(ID_NUM1, 462, spd);
73    Dxl.setPosition(ID_NUM2, 612, (spd));
74    Dxl.setPosition(ID_NUM3, 612, spd);
75    Dxl.setPosition(ID_NUM4, 562, spd);
76    Dxl.setPosition(ID_NUM5, 612, (spd));
77    Dxl.setPosition(ID_NUM6, 612, spd);
78    Dxl.setPosition(ID_NUM7, 562, spd);
79    Dxl.setPosition(ID_NUM8, 412, (spd));
80    Dxl.setPosition(ID_NUM9, 412, spd);
81    Dxl.setPosition(ID_NUM10, 462, spd);
82    Dxl.setPosition(ID_NUM11, 412, (spd));
83    Dxl.setPosition(ID_NUM12, 412, spd);
84  }
85
86
```

BAB 6 PENGUJIAN DAN ANALISIS

Bab pengujian dan analisi ini memuat hasil pengujian dan analisis terhadap sistem yang telah diimplementasikan pada bab sebelumnya.

6.1 Pengujian Sensor Srf08

Pada pengujian yang pertama yaitu dilakukan pengujian terhadap sensor jarak Srf08.

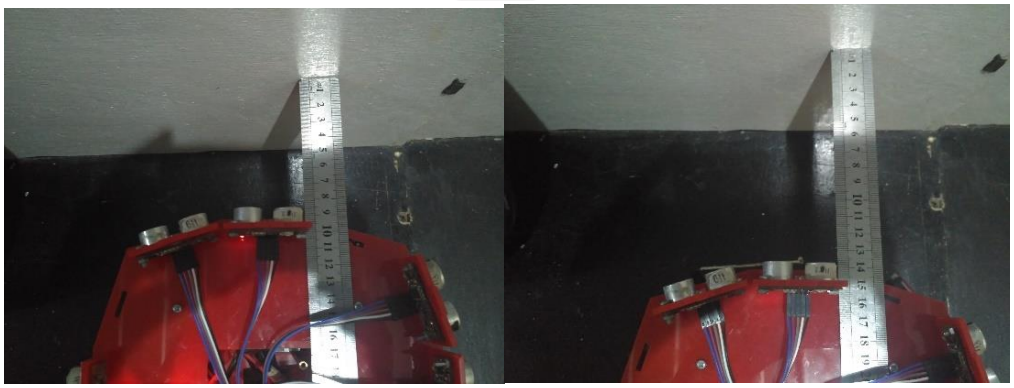
6.1.1 Tujuan Pengujian

Tujuan dari pengujian sensor Srf08 ini adalah untuk mengetahui tingkat akurasi jarak yang terbaca pada sensor mikrokontroller Arduino Uno terhadap jarak sesungguhnya.

6.1.2 Prosedur Pengujian

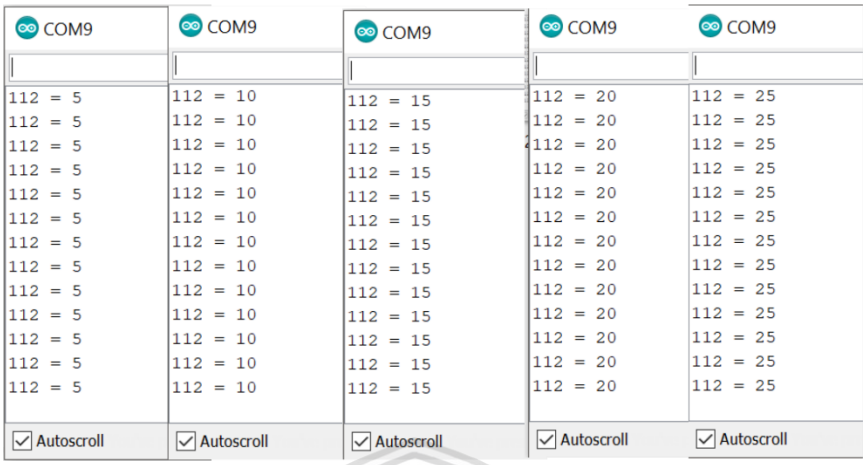
Prosedur pengujian dilakukan dalam beberapa langkah sebagai berikut:

1) Menyalakan mikrokontroller Arduino Uno dan Sensor Srf08
2) Jalankan kode program yang telah dibuat dan *upload* ke mikrokontroller Arduino Uno
3) Posisikan sensor berhadapan dengan dinding penghalang dan ukur menggunakan dengan penggaris jarak sensor dengan dinding tersebut seperti pada Gambar 6.1
4) Pengukuran dilakukan pada jarak yang terukur pada penggaris adalah jarak sensor dan dinding sejauh 5cm, 10cm, 15cm, 20cm dan 25cm
5) Lakukan pembacaan pada serial monitor dan melakukan pencatatan hasil pengujian
6) Langkah 1 sampai 5 dilakukan berulang sebanyak 5 sensor



Gambar 6.1 Prosedur Pengujian Sensor

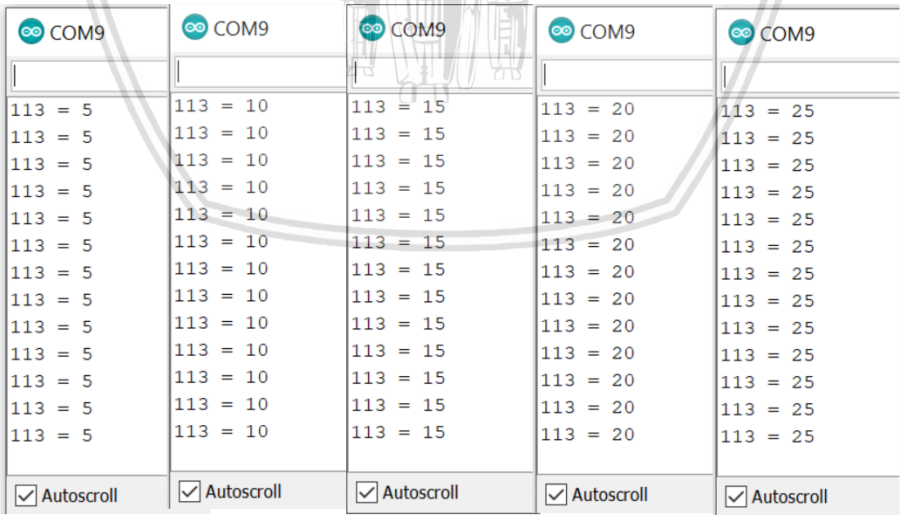
6.1.3 Hasil Pengujian



Gambar 6.2 Hasil Pengujian Sensor 1

Tabel 6.1 Pengujian sensor 1

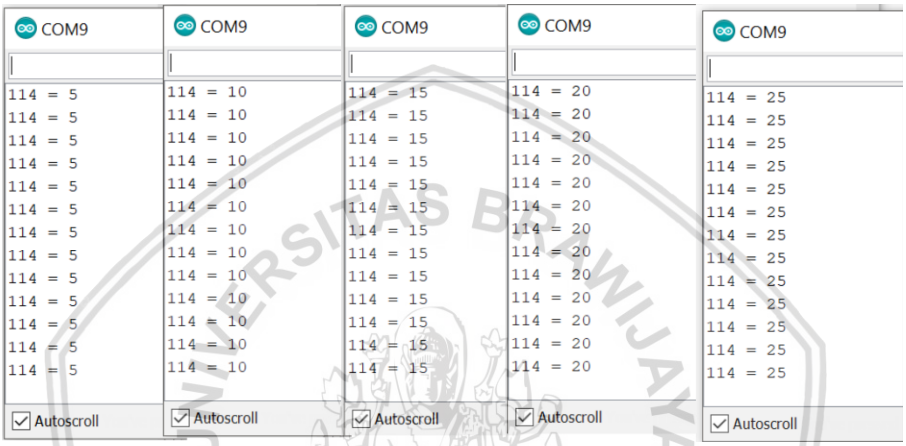
Nomor	Jarak Pada Penggaris	Jarak yang Terbaca	Error%
1	5 cm	5	0%
2	10 cm	10	0%
3	15 cm	15	0%
4	20 cm	20	0%
5	25cm	25	0%



Gambar 6.3 Hasil Pengujian Sensor 2

Tabel 6.2 Pengujian sensor 2

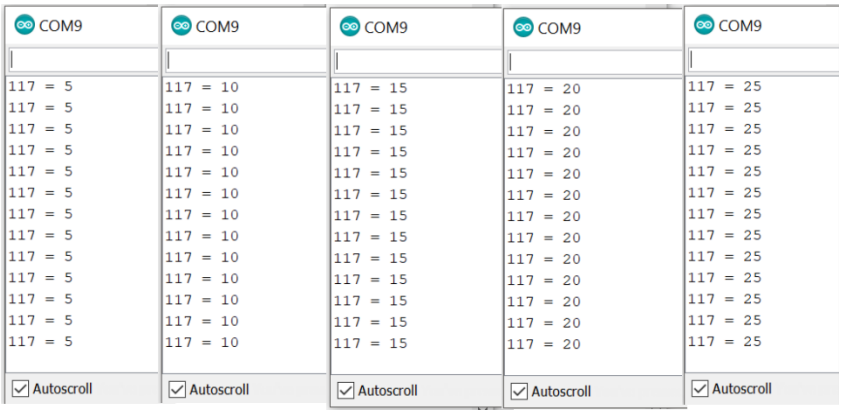
Nomor	Jarak Pada Penggaris	Jarak yang Terbaca	Error%
1	5 cm	5	0%
2	10 cm	10	0%
3	15 cm	15	0%
4	20 cm	20	0%
5	25cm	25	0%



Gambar 6.4 Hasil Pengujian Sensor 3

Tabel 6.3 Pengujian sensor 3

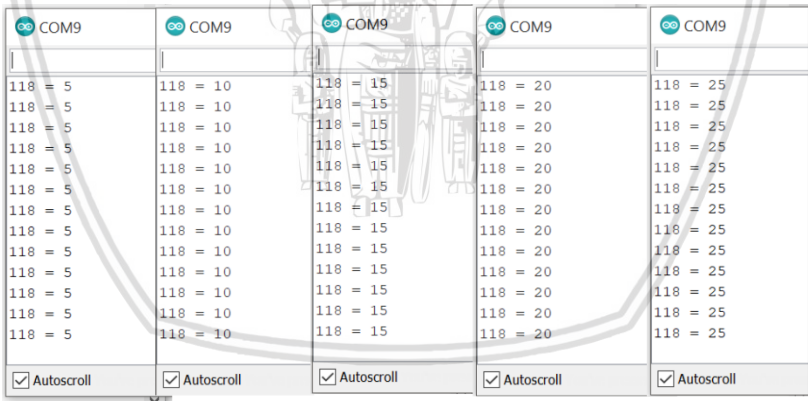
Nomor	Jarak Pada Penggaris	Jarak yang Terbaca	Error%
1	5 cm	5	0%
2	10 cm	10	0%
3	15 cm	15	0%
4	20 cm	20	0%
5	25cm	25	0%



Gambar 6.5 Hasil Pengujian Sensor 4

Tabel 6.4 Pengujian sensor 4

Nomor	Jarak Pada Penggaris	Jarak yang Terbaca	Error%
1	5 cm	5	0%
2	10 cm	10	0%
3	15 cm	15	0%
4	20 cm	20	0%
5	25cm	25	0%



Gambar 6.6 Hasil Pengujian Sensor 5

Tabel 6.5 Pengujian sensor 5

Nomor	Jarak Pada Penggaris	Jarak yang Terbaca	Error%
1	5 cm	5	0%
2	10 cm	10	0%
3	15 cm	15	0%
4	20 cm	20	0%
5	25cm	25	0%

6.1.4 Analisis Hasil Pengujian

Dari data yang diperoleh berdasarkan hasil pengujian melalui, didapatkan bahwa tingkat *error* pembacaan sensor Srf08 yang telah dilakukan pada pengujian sensor 1 sampai dengan sensor 5 adalah 0%. Atau dapat dikatakan tingkat akurasi sensor Srf08 adalah sebesar 100%. Untuk mengetahui tingkat *error* digunakan persamaan sebagai berikut:

$$Sistem\ Error(\%) = \frac{PL-PT}{PL} x\ 100$$

Keterangan:

PL: Pembacaan nilai sensor

PT: Pembacaan pada penggaris

6.2 Pngujian Manuver Robot

Pada pengujian yang kedua dilakukan perbandingan antara manuver robot *omni direction* dengan manuver robot memutar.

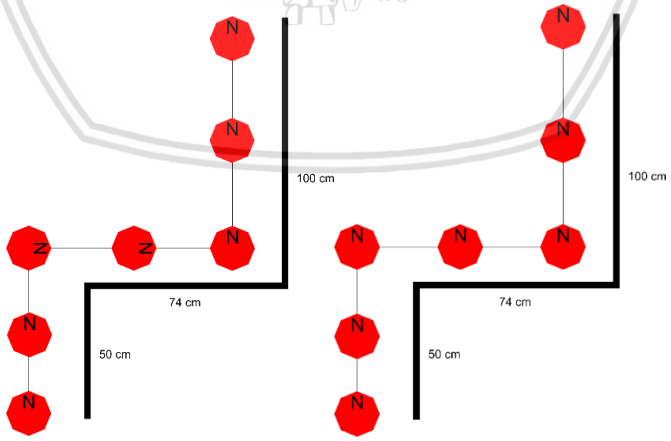
6.2.1 Tujuan Pengujian

Mengetahui perbandingan kecepatan antara manuver robot *omni direction* dengan manuver robot memutar.

6.2.2 Prosedur Pengujian

Prosedur pengujian dilakukan dalam beberapa langkah sebagai berikut:

1) Membuat lintasan robot sejauh 224cm dimana ditengahnya terdapat kondisi lintasan berbelok kearah kanan dan kiri yang digambarkan pada Gambar 6.7



Gambar 6.7 Lintasan Pengujian Manuver Robot

2) Nyalakan robot dan jalankan robot dengan program manuver robot dengan memutar

3) Kecepatan diukur menggunakan *stopwatch* dimulai pada saat robot berdiri dan selesai pada saat robot melewati batas akhir lintasan

4) Catat hasil kecepatan manuver robot dengan memutar pada tabel hasil pengujian
5) Pengujian dilakukan sebanyak 10 kali percobaan
6) Ulangi pengujian dengan program manuver robot *omni direction* sebanyak 10 kali percobaan

6.2.3 Hasil Pengujian

Tabel 6.6 Hasil Pengujian Manuver Robot

Pengujian Ke	Kecepatan manuver robot memutar (detik)	Kecepatan anuver robot *omni direction* (detik)
1	00:52.8	00:45.1
2	00:49.5	00:45.0
3	00:48.8	00:44.8
4	00:49.2	00:44.0
5	00:46.0	00:44.0
6	00:45.4	00:44.0
7	00:49.9	00:43.6
8	00:45.5	00:44.5
9	00:50.5	00:44.0
10	00:45.8	00:44.3
Rata-rata	00:48.4	00:44.3

6.2.4 Analisis Hasil Pengujian

Dari data yang diperoleh berdasarkan tabel hasil pengujian, dapat diketahui bahwa kecepatan manuver robot *omni direction* lebih cepat dibandingkan manuver robot memutar dengan selisih rata-rata 4.1 detik. Dari data uji juga terlihat bahwa waktu yang ditempuh pada robot manuver *omni direction* lebih stabil karena perbedaan waktu yang tidak terlalu jauh disetiap pegujian dengan nilai tercepat adalah 44.0 detik dan yang paling lama adalah 45.1 detik yang berarti selisih maksimumnya adalah 1.1 detik. Berbeda dengan robot yang bermanuver dengan memutar badan robot dimana waktu tercepat yang tercatat adalah 45.4 detik dan waktu yang paling lama adalah 52.8 detik dengan selisih 7.4 detik.

6.3 Pengujian Nilai Derajat Keanggotaan Input *Fuzzy* Sensor Depan

Pada pengujian ini dilakukan pencarian nilai keanggotaan *fuzzy* yang dapat diimplementasikan pada sistem.

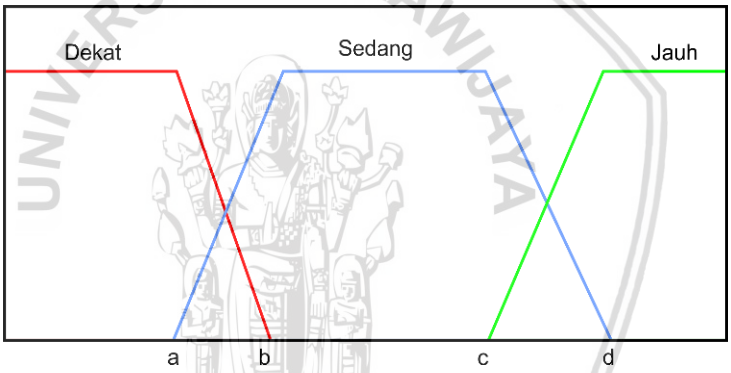
6.3.1 Tujuan Pengujian

Mengetahui nilai keanggotaan input *fuzzy* sensor depan yang dapat di implementasikan pada sistem

6.3.2 Prosedur Pengujian

Prosedur pengujian dilakukan dalam beberapa langkah sebagai berikut:

1) Menyalakan dan menjalankan robot pada lintasan robot
2) Menjalankan kode program yang telah dibuat
3) Merubah nilai a, b, c dan d ke nilai yang lebih besar ataupun ke nilai yang lebih kecil yang tergmbar pada Gambar 6.8. Kemudian *upload* kode program tersebut



Gambar 6.8 Grafik Fungsi Nilai Keanggotaan Sensor Depan

4) Jalankan kembali robot pada lintasan dan amati gerakan dan manuver robot kemudian catat hasil pengamatan pada tabel hasil pengujian

6.3.3 Hasil Pengujian

Tabel 6.7 Hasil Pengujian Fungsi Nilai Keanggotaan Sensor Depan

Nomor	a	b	c	d	Hasil
1	6	9	25	28	Menabrak
2	7	10	26	29	Manuver Sukses
3	8	11	27	30	Manuver Sukses
4	11	14	30	33	Manuver Sukses
5	12	15	31	34	Manuver Sukses

6.3.4 Analisis Hasil Pengujian

Dari data hasil pengujian diketahui bahwa pergeseran nilai fungsi keanggotaan sensor depan dapat di implementasikan dengan batasan nilai a antara 7 cm– 12 cm, nilai b antara 9cm – 15cm, c bernilai 25cm-31cm dan d bernilai 28cm-34cm. Robot akan dapat menabrak dinding depan jika nilai a kurang dari 7cm dan manuver robot sukses jika nilai a lebih dari sama dengan 7cm.

6.4 Pengujian Nilai Derajat Keanggotaan Input *Fuzzy* Sensor Kanan dan Sensor Miring

Pada pengujian ini dilakukan pencarian nilai keanggotaan *fuzzy* yang dapat diimplementasikan pada sistem.

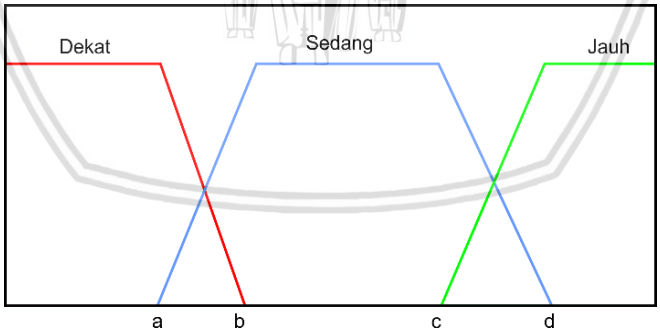
6.4.1 Tujuan Pengujian

Mengetahui nilai keanggotaan input *fuzzy* sensor kanan dan sensor miring yang dapat di implementasikan pada sistem. Sensor kanan dan sensor miring

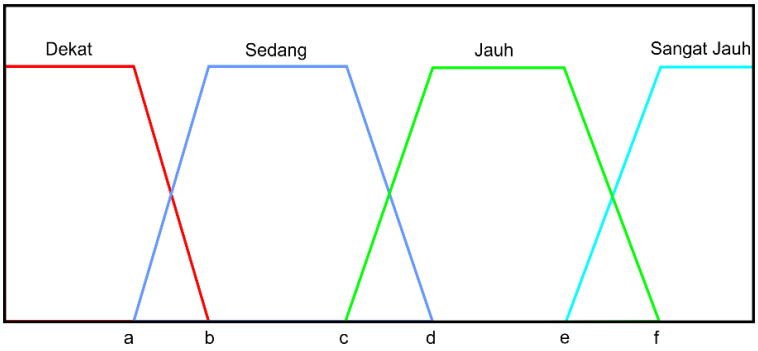
6.4.2 Prosedur Pengujian

Prosedur pengujian dilakukan dalam beberapa langkah sebagai berikut:

1) Menyalakan dan menjalankan robot pada lintasan robot
2) Menjalankan kode program yang telah dibuat
3) Merubah nilai a, b, c dan d ke pada sensor kanan dan nilai a, b, c, d, e dan f pada sensor miring kenilai yang lebih besar ataupun ke nilai yang lebih kecil yang tergmbar pada Gambar 6.9 dan Gambar 6.10. Kemudian *upload* kode program tersebut



Gambar 6.9 Grafik Fungsi Nilai Keanggotaan Sensor Kanan



Gambar 6.10 Grafik Fungsi Nilai Keanggotaan Sensor Miring

4) Jalankan kembali robot pada lintasan dan amati gerakan dan manuver robot kemudian catat hasil pengamatan pada table hasil pengujian

6.4.3 Hasil Pengujian

Tabel 6.8 Hasil Pengujian Nilai Keanggotaan Sensor Kanan dan Miring

No.	Kanan				Miring						Hasil
	a	b	c	d	a	b	c	d	e	f	
1	3	6	8	11	4	7	9	12	16	21	Manuver sukses
2	4	7	9	12	5	8	10	13	17	22	Manuver sukses
3	7	10	12	15	8	11	13	16	20	25	Manuver sukses
4	15	18	20	23	16	19	21	24	28	33	Manuver sukses
5	16	19	21	24	17	20	22	25	29	34	Manuver sukses

6.4.4 Analisis Hasil Pengujian

Dari data hasil pengujian diketahui bahwa dari lima pergeseran nilai fungsi keanggotaan sensor kanan dan sensor miring yang dilakukan memiliki hasil manuver sukses. Dari lima pengujian yang dilakukan kelima-limanya berhasil dijalankan oleh sistem. Akan tetapi ada batasan pada jarak robot dengan dinding yang harus dipenuhi jika sistem diimplementasikan pada arena kontes robot pemadam api dikarenakan dimensi lintasan jalan robot adalah selebar 46cm.

6.5 Pengujian Metode *Fuzzy*

Pada pengujian ini dilakukan pengujian metode *fuzzy* dengan cara membandingkan hasil implementasi *fuzzy* dengan perhitungan secara manual

6.5.1 Tujuan Pengujian

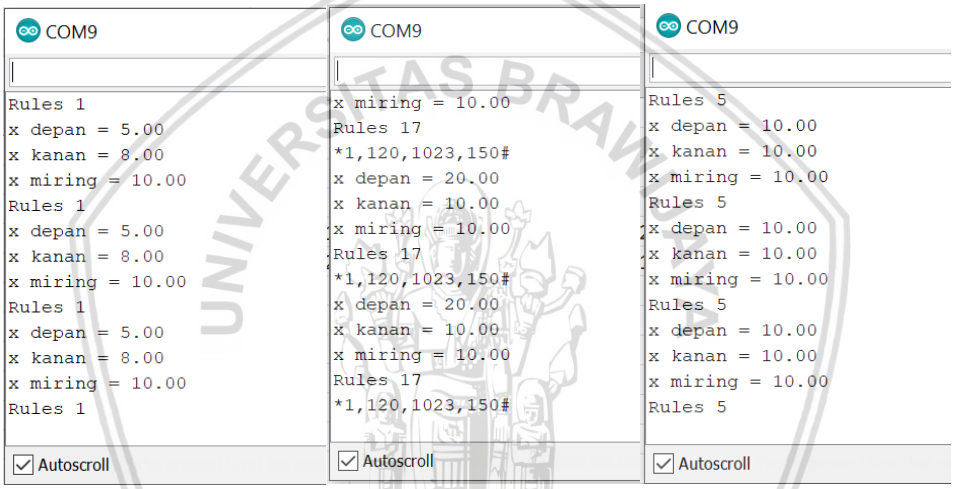
Mengetahui hasil perbandingan antara hasil implementasi *fuzzy* dengan perhitungan manual sehingga akan didapatkan tingkat error pada sistem.

6.5.2 Prosedur Pengujian

Prosedur pengujian dilakukan dalam beberapa langkah sebagai berikut:

1) Menjalankan robot dengan program input nilai sensor sebagai nilai derajat keanggotaan input *fuzzy* secara manual dengan nilai *random*
2) Melakukan pencatatan hasil perhitungan *fuzzy* sesuai dengan data pada serial monitor
3) Melakukan percobaan sebanyak 10 kali dengan nilai input sensor depan, sensor kanan dan sensor miring secara *random*
4) Melakukan perhitungan secara manual dan membandingkan hasilnya dengan hasil implementasi

6.5.3 Hasil Pengujian



Gambar 6.11 Hasil Pengujian Metode *Fuzzy*

Tabel 6.9 Hasil Pengujian Metode *Fuzzy*

No	Sensor Depan	Sensor Kanan	Sensor miring	Hasil Implementasi	Hasil Perhitungan Manual
1	5	8	10	Rule1 Manuver Kiri	Rule 1 Manuver Kiri
2	20	10	10	Rule 17 Kanan Cepat, Kiri Cepat, Speed pelan	Rule 17 Kanan Cepat, Kiri Cepat, Speed pelan
3	10	10	10	Rule 5 Manuver Kiri	Rule 5 Manuver Kiri
4	15	15	5	Rule 20 Kanan Pelan, Kiri Cepat, Speed Pelan	Rule 20 Kanan Pelan, Kiri Cepat, Speed Pelan
5	20	13	14	Rule 16	Rule 16

				Kanan Cepat, Kiri Pelan, Speed Pelan	Kanan Cepat, Kiri Pelan, Speed Pelan
6	15	8	9	Rule 12 Kanan Cepat, Kiri Pelan, Speed Pelan	Rule 12 Kanan Cepat, Kiri Pelan, Speed Pelan
7	30	8	9	Rule 24 Kanan Cepat, Kiri Pelan, Speed Cepat	Rule 24 Kanan Cepat, Kiri Pelan, Speed Cepat
8	30	10	11	Rule 29 Kanan Cepat, Kiri Cepat, Speed Cepat	Rule 29 Kanan Cepat, Kiri Cepat, Speed Cepat
9	30	9	10	Rule 29 Kanan Cepat, Kiri Cepat, Speed Cepat	Rule 29 Kanan Cepat, Kiri Cepat, Speed Cepat
10	27	9	14	Rule 17 Kanan Cepat, Kiri Cepat, Speed Pelan	Rule 17 Kanan Cepat, Kiri Cepat, Speed Pelan

Perhitungan secara manual:

Diketahui:

Sensor depan = 5

Sensor kanan = 8

Sensor miring = 10

1. Fuzzifikasi

a. Derajat keanggotaan sensor depan

Dekat : 1

Sedang : 0

Jauh : 0

b. Derajat keanggotaan sensor kanan

Dekat : $\frac{10-8}{10-7}$ = 0.67

Sedang : $\frac{8-7}{10-7}$ = 0.33

Jauh : 0

c. Derajat keanggotaan sensor miring

Dekat : $\frac{11-10}{11-8}$ = 0.33

Sedang : $\frac{10-8}{11-8}$ = 0.67

Jauh : 0

Sangat Jauh : 0

2. Inferensi

Rules [0]: IF (DEPAN DEKAT && KANAN DEKAT && MIRING DEKAT)

α = min (μDEPAN DEKAT ∩ μKANAN DEKAT ∩ μMIRING DEKAT)

= min (1 ∩ 0.67 ∩ 0.33)

= 0.33

Rules [1]: IF (DEPAN DEKAT && KANAN DEKAT && MIRING SEDANG)

α = min (μDEPAN DEKAT ∩ μKANAN DEKAT ∩ μMIRING SEDANG)

= min (1 ∩ 0.67 ∩ 0.67)

= 0.67

Rules [4]: IF (DEPAN DEKAT && KANAN SEDANG && MIRING DEKAT)

α = min (μDEPAN DEKAT ∩ μKANAN SEDANG ∩ μMIRING DEKAT)

= min (1 ∩ 0.33 ∩ 0.33)

= 0.33

3. Defuzzifikasi

Output = max (rules[0], rules[1], rules[2], rules[3], rules[4], rules[5], rules[6], rules[7], rules[8], rules[9], rules[10], rules[11], rules[12], rules[13], rules[14], rules[15], rules[16], rules[17], rules[18], rules[19], rules[20], rules[21], rules[22], rules[23], rules[24], rules[25], rules[26], rules[27], rules[28], rules[29], rules[30], rules[31], rules[32], rules[33], rules[34], rules[35])

Output = max (0.33, 0.67, 0, 0, 0.33, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Output = Rules [1] = manuver kiri

6.5.4 Analisis Hasil Pengujian

Dari data hasil pengujian pada Tabel 6.10 didapatkan bahwa dari 10 kali percobaan hasil output impementasi dan hasil perhitungan secara manual didapatkan hasil yang sama antara hasil implementasi dengan perhitungan secara manual. Sehingga dapat disimpulkan bahwa tingkat *error* pada implementasi logika *fuzzy* pada sistem adalah sebesar 0% yang berarti logika *fuzzy* berjalan pada sistem dengan akurasi sebesar 100%.

BAB 7 KESIMPULAN DAN SARAN

Pada bab ini membahas tentang kesimpulan dan saran yang diperoleh dari hasil pengujian dan analisis yang telah dilakukan.

7.1 Kesimpulan

Berdasarkan hasil dari tahapan perancangan, implementasi, pengujian serta analisis hasil pengujian yang telah dilakukan, penulis dapat menarik kesimpulan bahwa :

1. Robot *quadruped* berukuran simetris disetiap sisinya sehingga robot dapat berjalan ke empat arah yang berbeda dengan gerakan yang sama yang akan mempermudah dalam pembuatan *motion* robot. Bentuk robot *quadruped* adalah sebagaimana disebutkan pada bagian perancangan sistem.
2. Tingkat akurasi pembacaan sensor jarak Srf08 yang berjumlah 5 buah sensor berdasarkan hasil pengujian yang telah dilakukan dengan membandingkan nilai pembacaan sensor dengan alat ukur penggaris mempunyai tingkat akurasi mencapai 100% di semua sensor.
3. Metode *fuzzy* sugeno berhasil diimplementasikan pada algorita *wall following* dan robot dapat melakukan manuver *omni direction* sebagaimana dengan proses dimulai dari fuzzifikasi, inferensi dan defuzzifikasi sebagaimana dijelaskan pada bagian perancangan sistem dan telah dilakukan pengujian dengan tingkat *error* sebesar 0%.
4. Manuver robot tanpa memutar badan robot lebih cepat dibandingkan manuver dengan memutar badan robot dengan selisih rata-rata 4.1 detik. Manuver robot tanpa memutar badan robot memiliki kecepatan yang lebih konstan dibandingkan dengan manuver robot dengan memutar badan robot dikarenakan tidak menentunya kondisi robot pada saat melakukan putaran arah.

7.2 Saran

Berdasarkan kesimpulan yang telah didapatkan, terdapat beberapa saran untuk penelitian selanjutnya sebagai langkah pengembangan sistem ini, yaitu :

1. Disarankan untuk menggunakan sensor lain yang dapat dijalankan secara parallel tetapi harus mempunyai tingkat akurasi yang sama sehingga pembacaan sensor menjadi lebih cepat dan robot juga akan berjalan lebih cepat.
2. Disarankan menggunakan metode logika *fuzzy* dengan defuzzifikasi menggunakan metode centroid agar pergerakan kecepatan kakai robot dapat lebih dinamis sehingga jarak antara robot dengan dinding akan lebih presisi.

3. Disarankan membuat desain *body* kaki robot yang lebih baik dan menerapkan metode *inverse kinematic* pada pergerakan jalan robot agar dapat berjalan dengan lebih baik dan seimbang.

DAFTAR PUSTAKA

Arduino, 2016. *Arduino Uno REV3.* [Online] Available at: https://store.arduino.cc [Accessed 24 Agustus 2017].

Ari Azhar, K. D. K. W. H. S., 2015. Perancangan Fuzzy Logic Model Sugeno untuk Wall Tracking pada Robot Pemadam Api.

Budiharto W, S. D., 2014. *Artificial Intelegence Konsep dan Penerapannya.* Jakarta: CV. Andi Offset.

Fahmizal, 2013. Development of a Fuzzy Logic Wall Following Controller for Steering Mobile Robots. *Proceedings of 2013 International Conference on Fuzzy Theory and Its Application* .

KRI, 2017. [Online] Available at: https://www.kontesrobotindonesia.org/kri-2017/

Kui Qian, A. S., 2012. Autonomous navigation for mobile robot based on a sonar ring and its implementation. *IEEE Xplore Digital Library.*

Kusumadewi, S. &. P. H., 2013. *Aplikasi Logika Fuzzy untuk Pendukung Keputusan.* Yogyakarta: Graha Ilmu.

Rahman, M., 2016. Penerapan Metode Fuzzy Pada Robot Beroda Menggunakan Omni-Directional Wheels. *JUTISI,* Volume 5, p. 1022 – 1172 .

Ramadan, H., 2018. Scoring System Otomatis Pada Lomba Menembak Dengan Target Sillhoutte Hewan Menggunakan Logika Fuzzy. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume II, p. 9.

ROBOTIS, 2006. *User Manual Dynamixel AX-12,* s.l.: s.n.

ROBOTIS, 2010. *OpenCM9.04.* [Online] [Accessed 9 20 2017].

ROBOTIS, 2017. *OpenCM 485 Expantion Board.* [Online] Available at: http://www.robotis.us/opencm-485-expansion-boar/

Suharto, S. B., n.d. Sistem Navigasi Wall Following Robot KRPAI Divisi Berkaki Menggunakan Kontroller PID.

Wang, X., 2011. *Wall Following Algorithm for a Mobile Robot Using Extended Kalman Filter,* Alabama: Auburn University.

www.acroname.com, n.d. *Devantech SRF08 UltraSonic Ranger* , s.l.: s.n.

www.robot-electronics.co.uk, 2017. *www.robot-electronics.co.uk.* [Online] Available at: https://www.robot-electronics.co.uk/htm/srf08tech.html [Accessed 19 Oktober 2017].